ProTECT Result Comparisons

Introduction

```
#only run these if necessary
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("variants")
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.6     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

ProTECT normally runs a pipeline including both DNA and RNA. The DNA and RNA are used to:

- compute a consensus haplotype
- call *somatic* (only in the tumor) mutations

These are then used to calculate binding frequencies and look at the rankings of each mutation for potential immunotherapy.

If we want to run it with only RNA we then need to provide the consensus haplotype from clinician, and call the mutations only from the RNA.

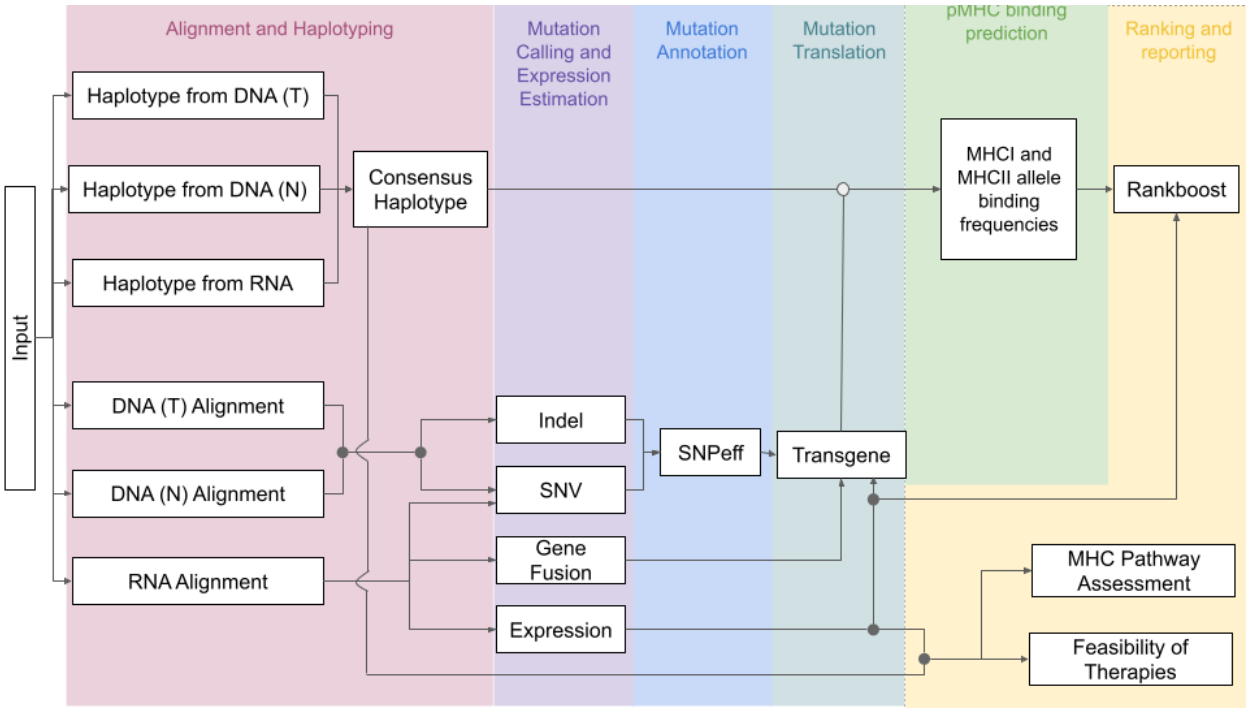
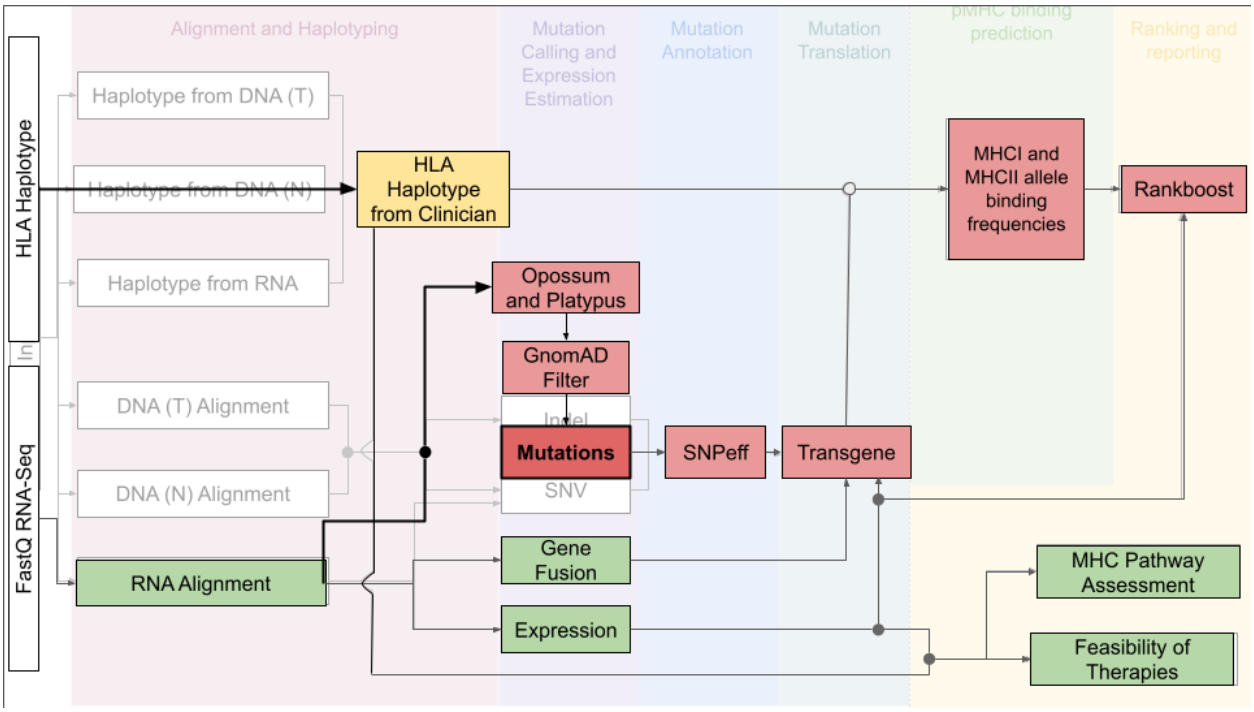


Figure 1: Diagram adapted from @raoProTECTPredictionTcell2019



This results in the changes shown in red. In order to validate this approach we want to confirm that the output of ProTECT when run on the same patient with only RNA rather than with just RNA and DNA is similar.

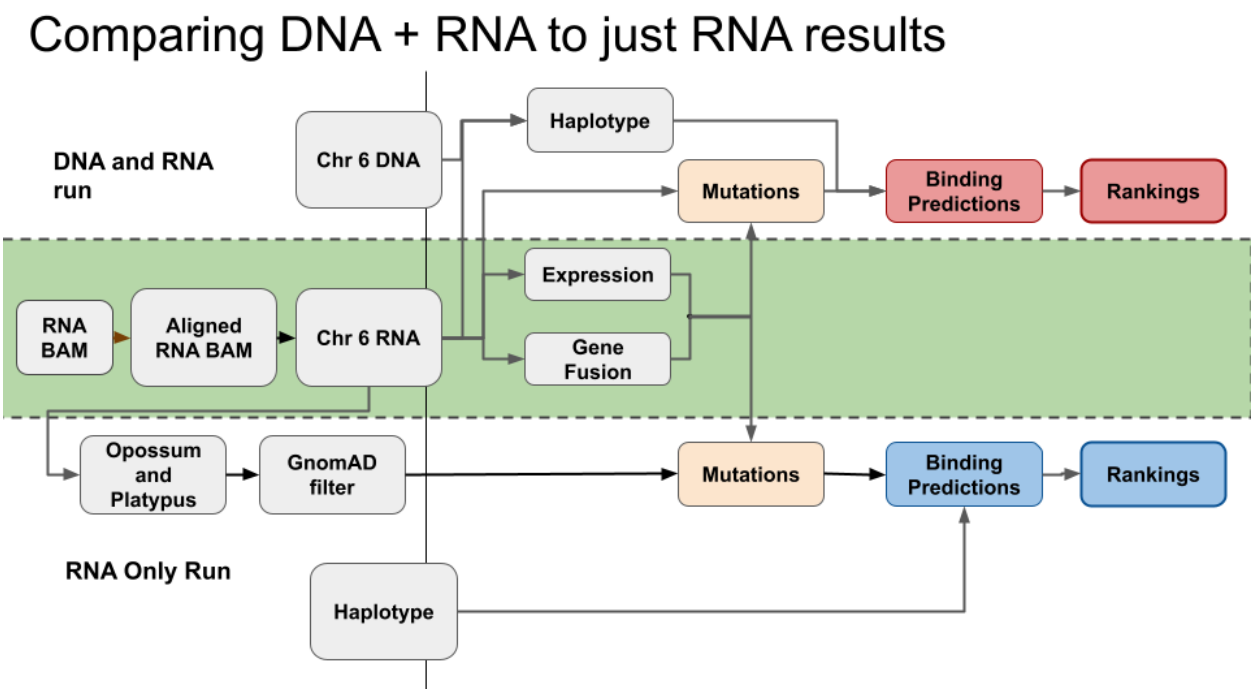


Figure 2: What we want to compare

Running ProTECT with DNA and RNA

Here we're performing analysis on DNA and RNA sequencing data from hcc1395 [@].

```
dir <- paste(getwd(),"fullbams", sep="/")
seq_data <- c("gerald_D1VCPACXX_6", "gerald_D1VCPACXX_1", "gerald_C1TD1ACXX_8_ACAGTG")
url <- "https://xfer.genome.wustl.edu/gxfer1/project/gms/testdata/bams/hcc1395_1percent"

for(bam in seq_data){
  file_loc <- paste0(dir, "/", bam, ".bam")
  if(file.exists(file_loc)){
    print(paste(bam, "exists."))
  }
  else {
    print(paste("Downloading", bam))
    download.file(paste0(url, "/", bam, ".bam"), paste0(dir, "/", bam, ".bam"))
  }
}
```

It was first prepared for a DNA and RNA ProTECT run by seperating in into fastq files using PicardTool's SamToFastq version 2.25.0

```
# Normal DNA
picard SamToFastq -I gerald_D1VCPACXX_6.bam -F gerald_D1VCPACXX_6_R1.fastq --SECOND_END_FASTQ gerald_D1
# Tumor DNA
picard SamToFastq -I /gerald_D1VCPACXX_1.bam -F /gerald_D1VCPACXX_1_R1.fastq --SECOND_END_FASTQ /gerald
# (Tumor) RNA
picard SamToFastq -I gerald_C1TD1ACXX_8_ACAGTG.bam -F gerald_C1TD1ACXX_8_ACAGTG_R1.fastq --SECOND_END_F
```

This could then be directly included into ProTECT's yaml input

```
patients:
    FULLBAM_NOVAR:
        tumor_dna_fastq_1: C:/Users/shellyes/Documents/protect-analysis/notebook/fullbams/gerald_D1VCPA
        tumor_dna_fastq_2: C:/Users/shellyes/Documents/protect-analysis/notebook/fullbams/gerald_D1VCPA
        normal_dna_fastq_1: C:/Users/shellyes/Documents/protect-analysis/notebook/fullbams/gerald_D1VCP
        normal_dna_fastq_2: C:/Users/shellyes/Documents/protect-analysis/notebook/fullbams/gerald_D1VCP
        tumor_rna_fastq_1: C:/Users/shellyes/Documents/protect-analysis/notebook/fullbams/gerald_C1TD1A
        tumor_rna_fastq_2: C:/Users/shellyes/Documents/protect-analysis/notebook/fullbams/gerald_C1TD1A
        tumor_type: 'SKCM'
```

ProTECT was then run successfully on the DNA and RNA.

Running ProTECT with RNA Only

Since our RNA has so many variants, for ProTECT to run effeciently the now aligned RNA bam was then separated into chromosomes:

```
samtools view rna_genome_sorted.bam -b chr${i} > rna_chr${i}.bam
```

Variant Calling

Currently we're running with Opossum and Platypus. This may not be best:

- Platypus stuck in Python2 and development seem to have moved on to Octopus with may be better regardless
- GATK Best Practices also has a suitable workflow without Opossum

However it's what we've done so far. For the workflow, we run Oppossum and Platypus on each bam file. This is all one big bash file in real life but seperated out, first we set up what we want to get:

```
# YAY INSTALLS
#yay -S pip samtools htslib

source venv/bin/activate
# check for RNA Tumor bam
homedir=$(pwd)
datadir=/scratch/drkthomp/fullbams
DATA=https://xfer.genome.wustl.edu/gxfer1/project/gms/testdata/bams/hcc1395
#RNA_TUMOR_BASE=gerald_C1TD1ACXX_8_ACAGTG
#RNA_TUMOR=/scratch/drkthomp/fullbams/${RNA_TUMOR_BASE}.sorted.bam
for i in {1..2}
do
RNA_TUMOR_BASE=rna_chr${i}
RNA_TUMOR=${datadir}/${RNA_TUMOR_BASE}.bam
#RNA_TUMOR_MD=/scratch/drkthomp/${RNA_TUMOR_BASE}md.bam
RNA_TUMOR_MD=${datadir}/callmd_${RNA_TUMOR_BASE}.bam
RNA_NORMAL=/scratch/drkthomp/fullbams/GCA_000001405.15_GRCh38_no_alt_analysis_set.fna
PICARD=picard.jar
```

Then we prep for opossum, so grab the picard and normal RNA if we need it, then index the bam. Note that these steps are functionally already done in ProTECT – ie, we are repeating things (suboptimal, reason to integrate)

```
echo "============"
echo "====chr${i}===="
echo "============"

if [ -f "$PICARD" ]; then
      echo "$PICARD exists."
    else
      wget https://github.com/broadinstitute/picard/releases/download/2.23.9/picard.jar $PICARD
fi

if [ -f "$RNA_NORMAL" ]; then
    echo "$RNA_NORMAL exists."
   else
    wget ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.15_GRCh38/seqs_for_alignm
    gunzip ${RNA_NORMAL}.gz
fi

echo "====CALMD"
if [ -f "$RNA_TUMOR_MD" ]; then
      echo "$RNA_TUMOR_MD exists."
    else
      samtools calmd ${RNA_TUMOR} ${RNA_NORMAL} > ${RNA_TUMOR_MD}
fi

echo "====INDEX"
if [ -f "${RNA_TUMOR}.bai" ]; then
      echo "${RNA_TUMOR}.bai exists."
    else
      samtools index -b $RNA_TUMOR
      echo "${RNA_TUMOR}.bai created"
fi
```

And then we actually run it:

```
echo "====OPOSSUM"
PROCESSED=${datadir}/opossum_${RNA_TUMOR_BASE}.bam
OPOSSUM=Opossum.py
if [ -f "$OPOSSUM" ]; then
    echo "$OPOSSUM exists."
  else
    wget -O $OPOSSUM https://raw.githubusercontent.com/BSGOxford/Opossum/master/Opossum.py
    2to3 -w $OPOSSUM
fi
# pip dependencies for opossum
pip install pysam

#woot
if [ -f "${PROCESSED}" ]; then
    echo "${PROCESSED} exists."
```

```
  else
    python $OPOSSUM --BamFile=$RNA_TUMOR --OutFile=$PROCESSED --SoftClipsExist=True
fi
```

Then we run Platypus. This is a little bit more difficult and is part of the reason I'm thinking of actually *not* running with Platypus.

```
# okay install HTSLib
HTSLIB_TAR=htslib-1.11
if [ -f "${HTSLIB_TAR}.tar.bz2" ]; then
    echo "${HTSLIB_TAR} exists."
  else
    wget -O ${HTSLIB_TAR}.tar.bz2 https://github.com/samtools/htslib/releases/download/1.11/htslib-1.11
    tar -xf ${HTSLIB_TAR}.tar.bz2
    cd $HTSLIB_TAR
    make install prefix=${homedir}/htslib
    # had to run these manually
fi

echo "====PLATYPUS"
export C_INCLUDE_PATH=${homedir}/htslib/include
export LIBRARY_PATH=${homedir}/htslib/lib
export LD_LIBRARY_PATH=${homedir}/htslib/lib
cd $homedir
PLATYPUS_DIR=platypus
PLATYPUS=${PLATYPUS_DIR}/bin/Platypus.py
if [ -f "$PLATYPUS" ]; then
    echo "$PLATYPUS exists."
  else
    git clone --depth=1 --branch=master https://github.com/andyrimmer/Platypus.git ${PLATYPUS_DIR}
    rm -rf ./${PLATYPUS_DIR}/.git
    cd ${PLATYPUS_DIR}
    make
fi
cd $homedir

#2to3 -w ${PLATYPUS_DIR}/
# first is suggested with opossum
python2 ~/miniconda3/envs/platypus/share/platypus-variant-0.8.1.2-0/Platypus.py callVariants --bamFiles=

done
```

Variant Filtering

and then subtract GnomAD from each platypus file:

```
bedtools subtract -header -a variants_opossum_rna_chr${i}.vcf -b /scratch/drkthomp/protect-index/gnomad/
```

Looking at Results

0 - Confirming Matches

We know that the RNA and DNA runs should match at some outputs.

Comparing Expressions

```
relIsoVals <-
  c("length",
    "effective_length",
    "expected_count",
    "TPM",
    "FPKM",
    "IsoPct")
toRun <-
  parse(text = paste0("getExpression('../", runsToCompare, "', 'isoforms')"))
for (i in seq_along(toRun)){
  assign(paste0("expressionIsoforms.", runsToCompare[i]), eval(toRun[[i]]))
}
```

Isoforms

```
## [1] "../FULLBAM_NOVAR/expression/rsem.isoforms.results"
## [1] "../FULLBAM_VARIANTS/expression/rsem.isoforms.results"
```

```
  isoforms <- mget(ls(pattern = "^expressionIsoforms.*")) %>%
  bind_rows(.id = "patients") %>% group_by(patients)
isoforms %>%  pivot_wider(names_from = patients, values_from = relIsoVals)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use ‘all_of(relIsoVals)‘ instead of ‘relIsoVals‘ to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
## # A tibble: 197,935 x 14
##    transcript_id gene_id length_expressi~ length_expressi~ effective_lengt~
##    <chr>         <chr>              <int>            <int>            <dbl>
##  1 ENST00000000~ ENSG00~             1103             1103             882.
##  2 ENST00000000~ ENSG00~             2756             2756            2535.
##  3 ENST00000000~ ENSG00~             2215             2215            1994.
##  4 ENST00000001~ ENSG00~             3732             3732            3511.
##  5 ENST00000001~ ENSG00~             4732             4732            4511.
##  6 ENST00000002~ ENSG00~             2176             2176            1955.
##  7 ENST00000002~ ENSG00~             2356             2356            2135.
##  8 ENST00000002~ ENSG00~             2079             2079            1858.
##  9 ENST00000002~ ENSG00~             8031             8031            7810.
## 10 ENST00000002~ ENSG00~             3802             3802            3581.
## # ... with 197,925 more rows, and 9 more variables:
## #   effective_length_expressionIsoforms.FULLBAM_VARIANTS <dbl>,
## #   expected_count_expressionIsoforms.FULLBAM_NOVAR <dbl>,
## #   expected_count_expressionIsoforms.FULLBAM_VARIANTS <dbl>,
```

```
## #   TPM_expressionIsoforms.FULLBAM_NOVAR <dbl>,
## #   TPM_expressionIsoforms.FULLBAM_VARIANTS <dbl>,
## #   FPKM_expressionIsoforms.FULLBAM_NOVAR <dbl>,
## #   FPKM_expressionIsoforms.FULLBAM_VARIANTS <dbl>,
## #   IsoPct_expressionIsoforms.FULLBAM_NOVAR <dbl>,
## #   IsoPct_expressionIsoforms.FULLBAM_VARIANTS <dbl>
```

```
isoforms %>% summarise(
  length = mean(length),
  effective_length = mean(effective_length),
  expected_count = mean(expected_count),
  TPM = mean(TPM),
  FPKM = mean(FPKM),
  IsPct = mean(IsoPct)
)
```

```
## # A tibble: 2 x 7
##   patients              length effective_length expected_count   TPM  FPKM IsPct
## * <chr>                  <dbl>            <dbl>          <dbl> <dbl> <dbl> <dbl>
## 1 expressionIsoforms.F~  1487.            1272.           506.  5.05  4.58  13.5
## 2 expressionIsoforms.F~  1487.            1272.           506.  5.05  4.58  13.5
```

```
relGeneVals <-
  c("length", "effective_length", "expected_count", "TPM", "FPKM")
toRun <-
  parse(text = paste0("getExpression('../", runsToCompare, "', 'genes')"))

for (i in seq_along(toRun))
  assign(paste0("expressionGenes.", runsToCompare[i]), eval(toRun[[i]]))
```

Genes

```
## [1] "../FULLBAM_NOVAR/expression/rsem.genes.results"
## [1] "../FULLBAM_VARIANTS/expression/rsem.genes.results"
```

```
genes <- mget(ls(pattern = "^expressionGenes.*")) %>%
  bind_rows(.id = "patients") %>% group_by(patients)
genes %>% summarise(
  length = mean(length),
  effective_length = mean(effective_length),
  expected_count = mean(expected_count),
  TPM = mean(TPM),
  FPKM = mean(FPKM)
)
```

```
## # A tibble: 2 x 6
##   patients                    length effective_length expected_count   TPM  FPKM
## * <chr>                        <dbl>            <dbl>          <dbl> <dbl> <dbl>
## 1 expressionGenes.FULLBAM_NO~  1395.            1190.          1726.  17.2  15.6
## 2 expressionGenes.FULLBAM_VA~  1395.            1190.          1726.  17.2  15.6
```

```
rm(genes)
rm(isoforms)
rm(mutations)
```

```
## Warning in rm(mutations): object 'mutations' not found
```

```
rm(expressionGenes.FULLBAM_NOVAR)
rm(expressionIsoforms.FULLBAM_NOVAR)
```

1 - Examining VCFs

Getting the VCFs

```
getVCF = function(name){
  chroms <- c(1,2,3,4,5,7,8,9)
  KVsep <- fixed(";")
Vsep <- fixed("=")
  mutations <- read.delim(paste(name, "vcf",sep="."), header=FALSE, comment.char="#")
  relevant <- c("CHROM",    "POS",   "ID",    "REF",   "ALT",   "QUAL", "FILTER",    "INFO")
  names(mutations) <- relevant
  return(mutations[,1:8] %>%
           mutate(mut_id = paste(CHROM, POS, REF, ALT, sep="_")) %>%
           mutate(rn = row_number()) %>%
  mutate(split = str_split(INFO, KVsep)) %>%
  unnest(split) %>%
  separate(split, into = c("key", "value"), Vsep) %>%
  spread(key, value) %>%
  select(-INFO) %>% select(-rn))
}
file_loc <- "~/igv-analysis/all_chromosomes"

RNA <- list()
DNA <- list()
chroms <- c(1,2,3,4,5,7,8,9,11,12,13,14,15,16,17,18,19,20,21,22)
chrom_names <- c(paste0("chr", chroms), "chr6")
```

There are three levels of VCF: #### the ones input into SNPeFF

```
RNA$input <- getVCF(paste0(file_loc, "/", "variants_chr1_subtracted"))
for (i in chroms) {
  RNA$input <- bind_rows(RNA$input,getVCF(paste0(file_loc, "/","variants_chr",i, "_subtracted")))
}
DNA$input <- getVCF(paste0(file_loc, "/","dna")) %>% filter(CHROM %in% chrom_names)
```

```
RNA$snpeffed <- getVCF("~/protect-analysis/CHR6_VARIANTS/mutations/snpeffed/mutations")
for (i in chroms) {
  print(paste("read snpeffed vcf for chromosome",i))
```

```
  RNA$snpeffed <- bind_rows(RNA$snpeffed,getVCF(paste0(file_loc, "/","chr",i, "_snpeffed")))
}


DNA$snpeffed <- getVCF(paste0(file_loc, "/","dna", "_snpeffed")) %>% filter(CHROM %in% chrom_names)
```

the onses which have been speffed

```

RNA$transgened <- getVCF("~/protect-analysis/CHR6_VARIANTS/mutations/transgened/mutations")
for (i in chroms) {
  print(paste("read transgened vcf for chromosome",i))
  RNA$transgened <- bind_rows(RNA$transgened,getVCF(paste0(file_loc, "/","chr",i, "_transgened")))
}
```

the transgene ones

```
## [1] "read transgened vcf for chromosome 1"
## [1] "read transgened vcf for chromosome 2"
## [1] "read transgened vcf for chromosome 3"
## [1] "read transgened vcf for chromosome 4"
## [1] "read transgened vcf for chromosome 5"
## [1] "read transgened vcf for chromosome 7"
## [1] "read transgened vcf for chromosome 8"
## [1] "read transgened vcf for chromosome 9"
## [1] "read transgened vcf for chromosome 11"
## [1] "read transgened vcf for chromosome 12"
## [1] "read transgened vcf for chromosome 13"
## [1] "read transgened vcf for chromosome 14"
## [1] "read transgened vcf for chromosome 15"
## [1] "read transgened vcf for chromosome 16"
## [1] "read transgened vcf for chromosome 17"
## [1] "read transgened vcf for chromosome 18"
## [1] "read transgened vcf for chromosome 19"
## [1] "read transgened vcf for chromosome 20"
## [1] "read transgened vcf for chromosome 21"
## [1] "read transgened vcf for chromosome 22"
```

```
DNA$transgened <- getVCF(paste0(file_loc, "/","dna", "_transgened")) %>% filter(CHROM %in% chrom_names)
```

We also want to load in the DNA with GnoMAD subtracted so we know what false negatives are due to that:

```
DNA$subtracted <- getVCF(paste0(file_loc, "/","dna", "_subtracted")) %>% filter(CHROM %in% chrom_names)
```

Transgened

```
mutations <- list(DNA = unique(c(filter(DNA$transgened, FILTER=="PASS")$mut_id)), RNA = unique(c(filter
gnomad <- unique(c(filter(DNA$transgened, !(mut_id %in% unique(c(DNA$subtracted$mut_id))))$mut_id))

transgene_touchup = function(vcf){
  return(vcf %>% mutate(QUAL = as.double(NA)) %>%  mutate(status = case_when(
  ((mut_id %in% mutations$RNA) & (mut_id %in% mutations$DNA)) ~ 'True Positive',
 (!(mut_id %in% mutations$RNA) & (mut_id %in% gnomad)) ~ 'False Negative - GnomAD',
  (!(mut_id %in% mutations$RNA) & (mut_id %in% mutations$DNA)) ~ 'False Negative',
 ((mut_id %in% mutations$RNA) & !(mut_id %in% mutations$DNA)) ~ 'False Positive',
  TRUE ~ 'True Negative'))%>%
   mutate(coverage = str_replace(coverage, "[.]", "")) %>%  mutate(coverage = str_replace(coverage, ","
  separate(coverage, into = c("supporting_reads", "overall_reads"), "/") %>% mutate(supporting_reads = a
}

examine <- bind_rows("RNA"=transgene_touchup(RNA$transgened), "DNA"=transgene_touchup(DNA$transgened),
```

Finding Truth Values

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 22828 rows [1, 2,
## 3, 4, 5, 6, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, ...].

## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion

## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 14660 rows [3, 6,
## 10, 11, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 30, 31, 50, 54, 55, 60, ...].

## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```

```
examine %>% group_by(status, src) %>%  summarize(count=n_distinct(mut_id))
```

Examine

```
## `summarise()` has grouped output by 'status'. You can override using the `.groups` argument.

## # A tibble: 8 x 3
## # Groups:   status [5]
##   status                   src    count
##   <chr>                    <chr>  <int>
## 1 False Negative           DNA      452
## 2 False Negative - GnomAD  DNA    12932
## 3 False Negative - GnomAD  RNA       44
## 4 False Positive           RNA   130813
## 5 True Negative            DNA    14701
## 6 True Negative            RNA    36539
## 7 True Positive            DNA      674
## 8 True Positive            RNA      674
```

```
filter(examine, FILTER=="PASS") %>% group_by(src, status) %>% summarize(count=n_distinct(mut_id))
```

```
## `summarise()` has grouped output by 'src'. You can override using the `.groups` argument.
```

```
## # A tibble: 5 x 3
## # Groups:   src [2]
##   src   status                   count
##   <chr> <chr>                    <int>
## 1 DNA   False Negative             452
## 2 DNA   False Negative - GnomAD    656
## 3 DNA   True Positive              674
## 4 RNA   False Positive          130813
## 5 RNA   True Positive              674
```

```
filter(examine, src=="RNA") %>% group_by(status) %>% summarize(count=n_distinct(mut_id),percent=n_distin
```

```
## # A tibble: 4 x 4
##   status                    count  percent coverage
## * <chr>                     <int>    <dbl>    <dbl>
## 1 False Negative - GnomAD     44 0.000262  NA
## 2 False Positive          130813 0.778      0.0826
## 3 True Negative            36539 0.217      0.703
## 4 True Positive              674 0.00401    0.152
```

```
filter(examine, src=="DNA") %>% group_by(status) %>% summarize(count=n_distinct(mut_id),percent=n_distin
```

```
## # A tibble: 4 x 4
##   status                   count percent coverage
## * <chr>                    <int>   <dbl>    <dbl>
## 1 False Negative             452  0.0157   0.0213
## 2 False Negative - GnomAD 12932  0.450    0
## 3 True Negative           14701  0.511    0
## 4 True Positive             674  0.0234   0.153
```

```
filter(examine, src=="RNA" & FILTER=="PASS") %>% group_by(status) %>% summarize(count=n_distinct(mut_id
```

```
## # A tibble: 2 x 4
##   status          count percent coverage
## * <chr>           <int>   <dbl>    <dbl>
## 1 False Positive 130813 0.995     0.0826
## 2 True Positive     674 0.00513   0.152
```

```
filter(examine, src=="DNA" &FILTER=="PASS") %>% group_by(status) %>% summarize(count=n_distinct(mut_id)
```

```
## # A tibble: 3 x 4
##   status                   count percent coverage
## * <chr>                    <int>   <dbl>    <dbl>
## 1 False Negative             452   0.254   0.0213
## 2 False Negative - GnomAD   656   0.368   0.05
## 3 True Positive             674   0.378   0.153
```

```
initial <- filter(examine,FILTER=="PASS") %>% group_by(src, status) %>% summarize(n=n_distinct(mut_id),
```

```
## `summarise()` has grouped output by 'src'. You can override using the `.groups` argument.
```

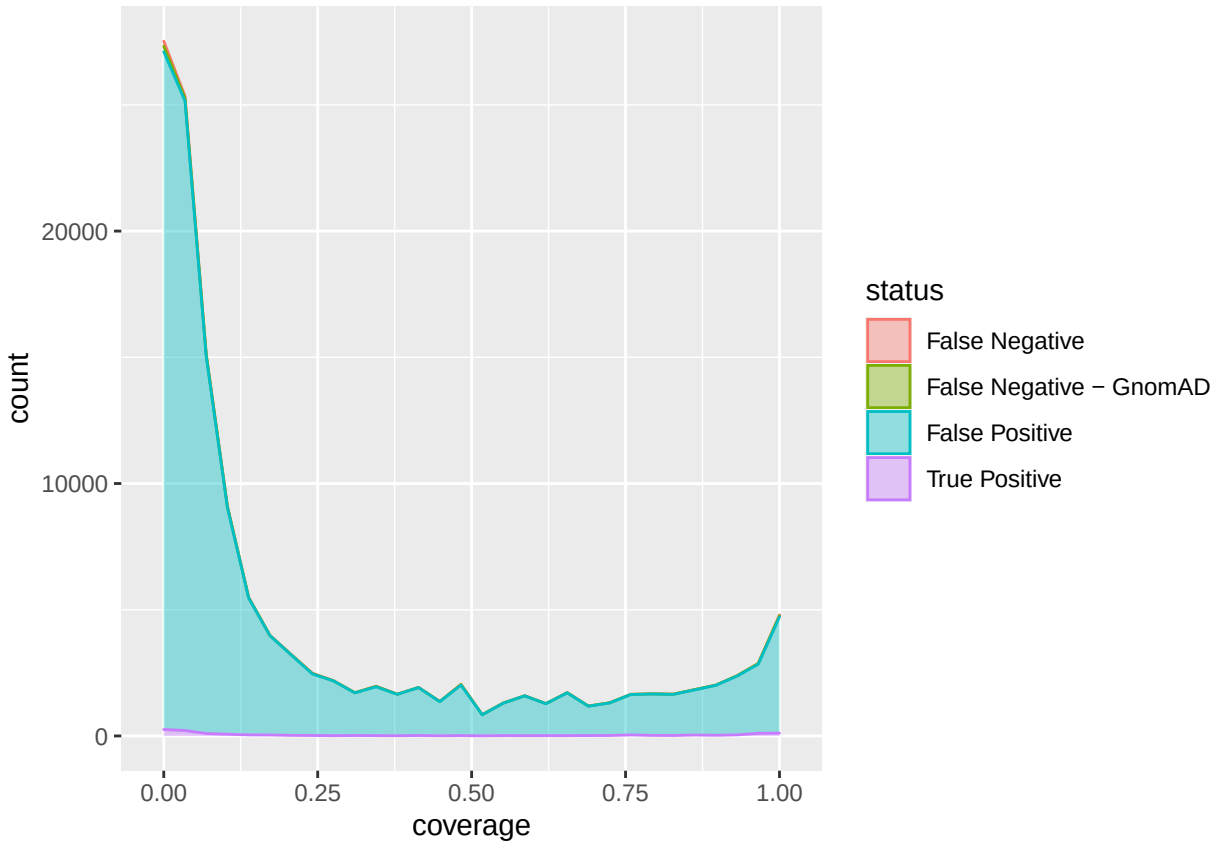
```
initial
```

```
## # A tibble: 5 x 4
## # Groups:   src [2]
##   src   status                      n coverage
##   <chr> <chr>                   <int>    <dbl>
## 1 DNA   False Negative            452   0.0213
## 2 DNA   False Negative - GnomAD   656   0.05
## 3 DNA   True Positive             674   0.153
## 4 RNA   False Positive          130813   0.0826
## 5 RNA   True Positive             674   0.152
```

```
area <- ggplot(filter(examine,FILTER=="PASS"), aes(coverage, fill=status,color=status)) + geom_area(sta
area
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

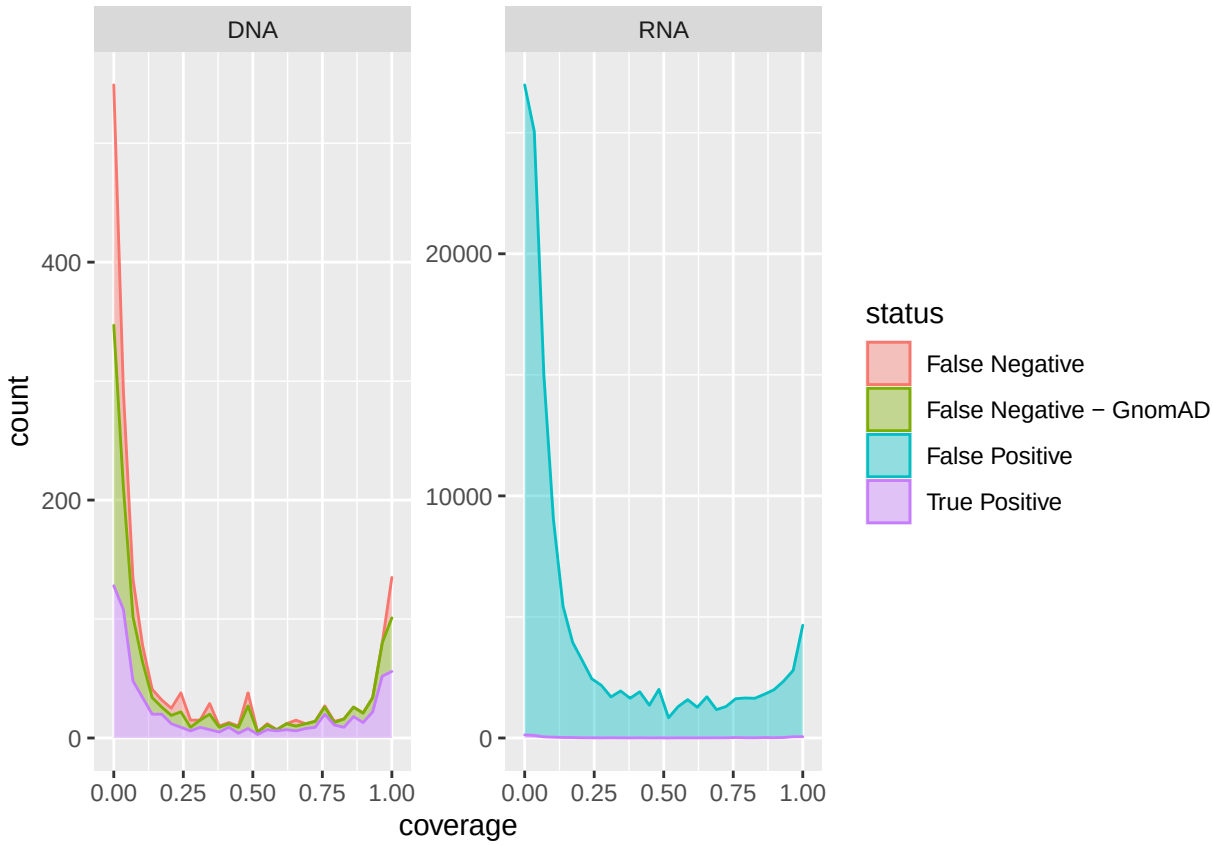
```
## Warning: Removed 33 rows containing non-finite values (stat_bin).
```



```
area + facet_wrap(vars(src),scales="free_y")
```

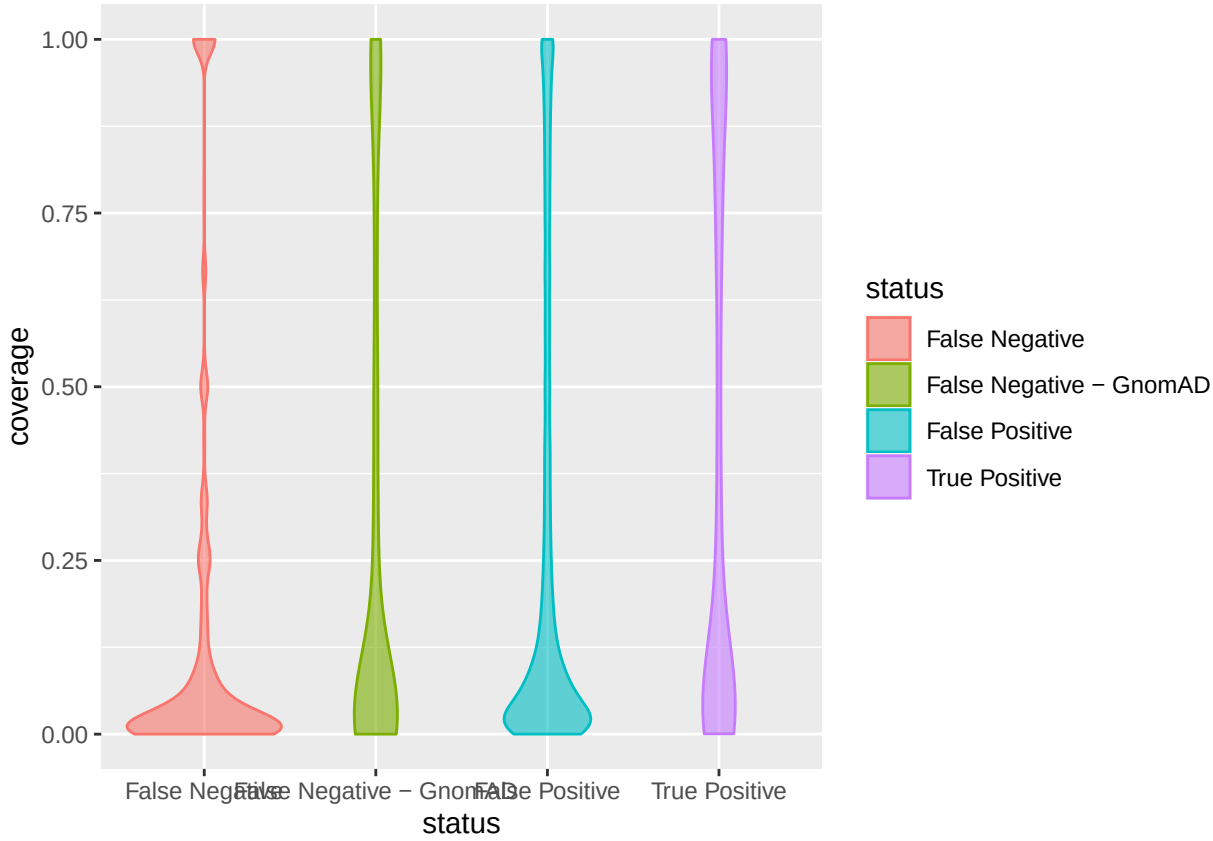
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 33 rows containing non-finite values (stat_bin).
```



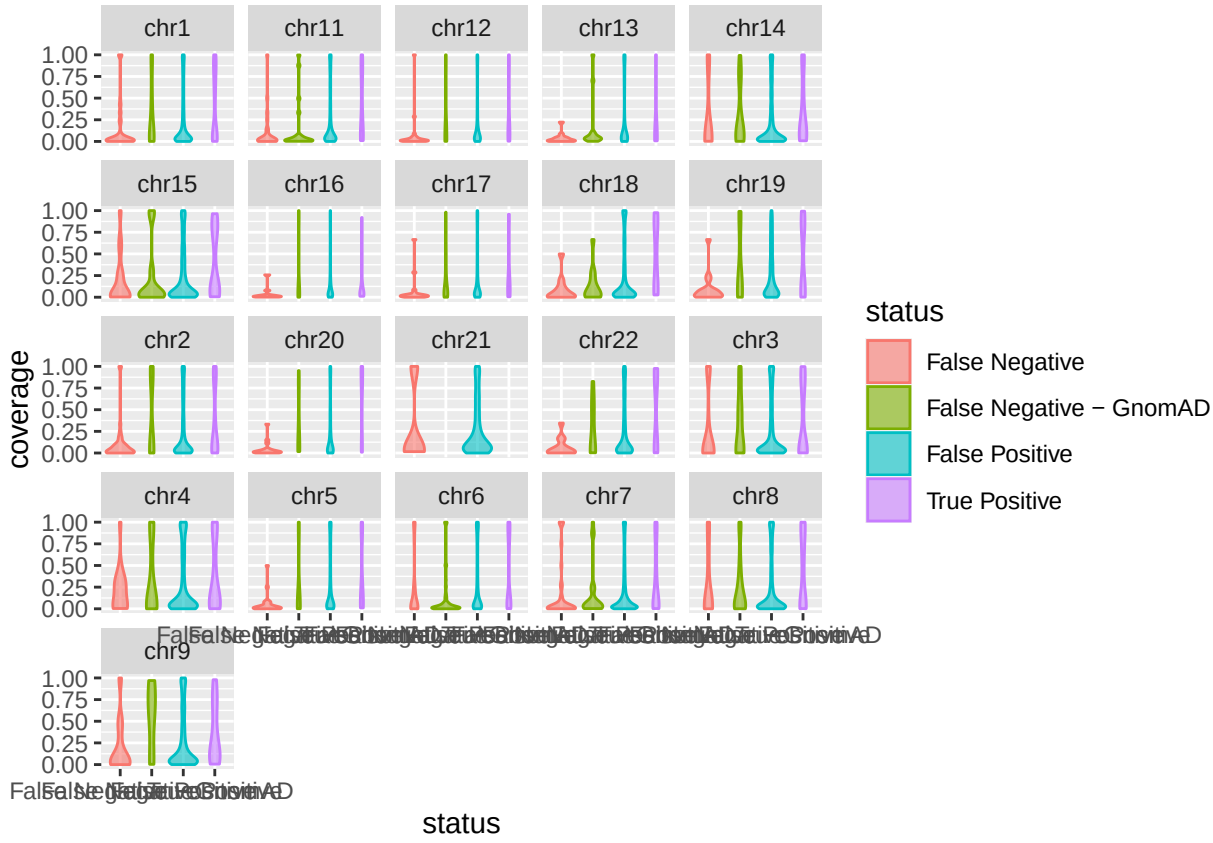

```
violin <- ggplot(filter(examine,FILTER=="PASS"), aes(status, coverage,fill=status, color=status)) + geo
violin
```

```
## Warning: Removed 33 rows containing non-finite values (stat_ydensity).
```



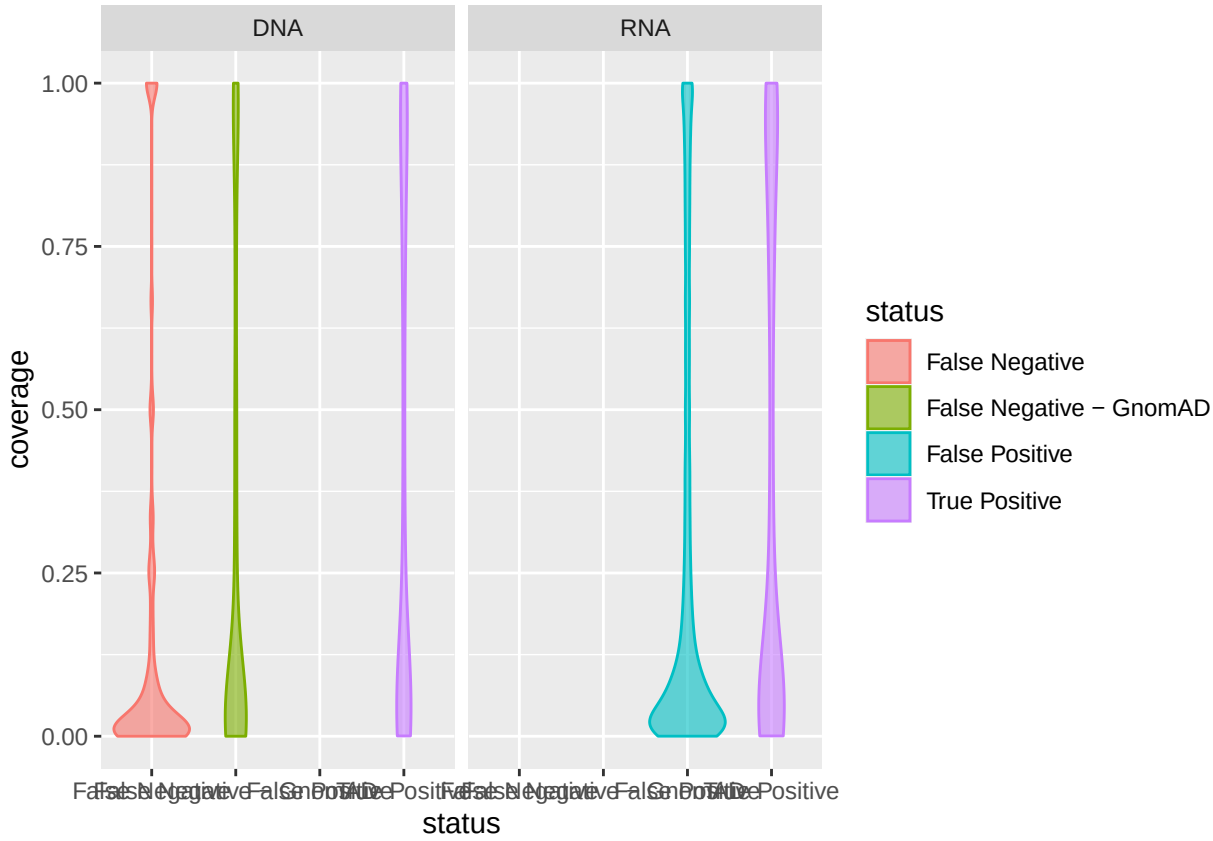

```
violin + facet_wrap(vars(CHROM))
```

```
## Warning: Removed 33 rows containing non-finite values (stat_ydensity).
```



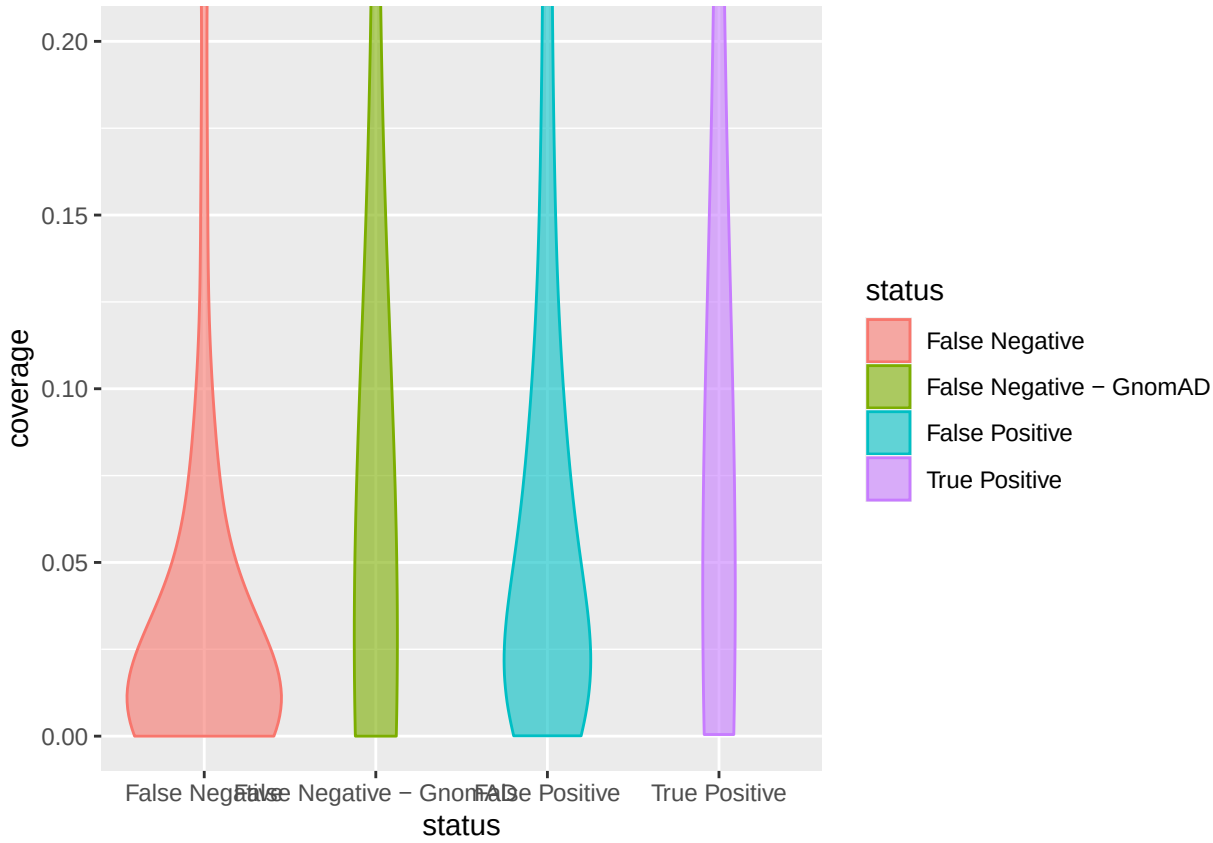

```
violin + facet_wrap(vars(src))
```

```
## Warning: Removed 33 rows containing non-finite values (stat_ydensity).
```



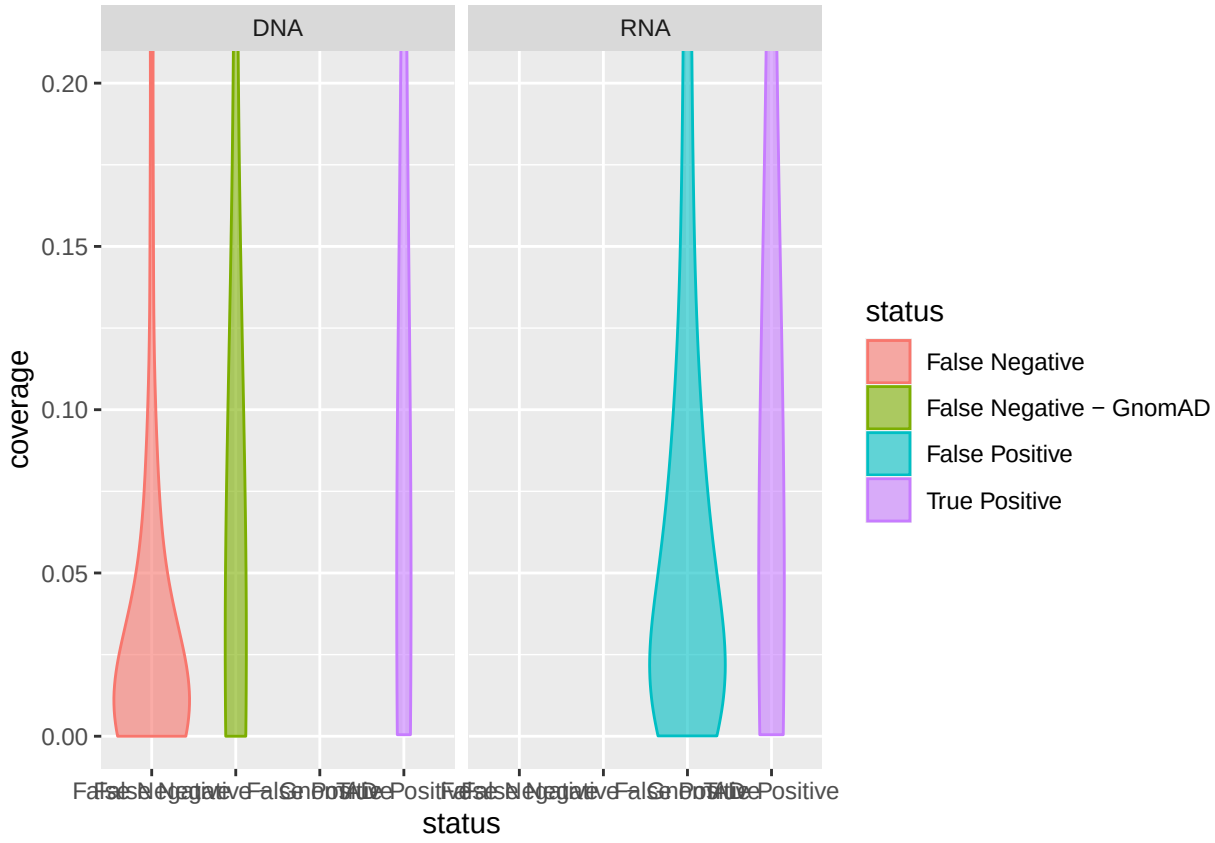

```
violin + coord_cartesian(ylim=c(0,.2))
```

```
## Warning: Removed 33 rows containing non-finite values (stat_ydensity).
```




```
violin + coord_cartesian(ylim=c(0,.2)) + facet_wrap(vars(src))
```

```
## Warning: Removed 33 rows containing non-finite values (stat_ydensity).
```




```
examine %>% filter(FILTER=="PASS" & status != "False Negative - GnomAD") %>% group_by(status) %>% summa
```

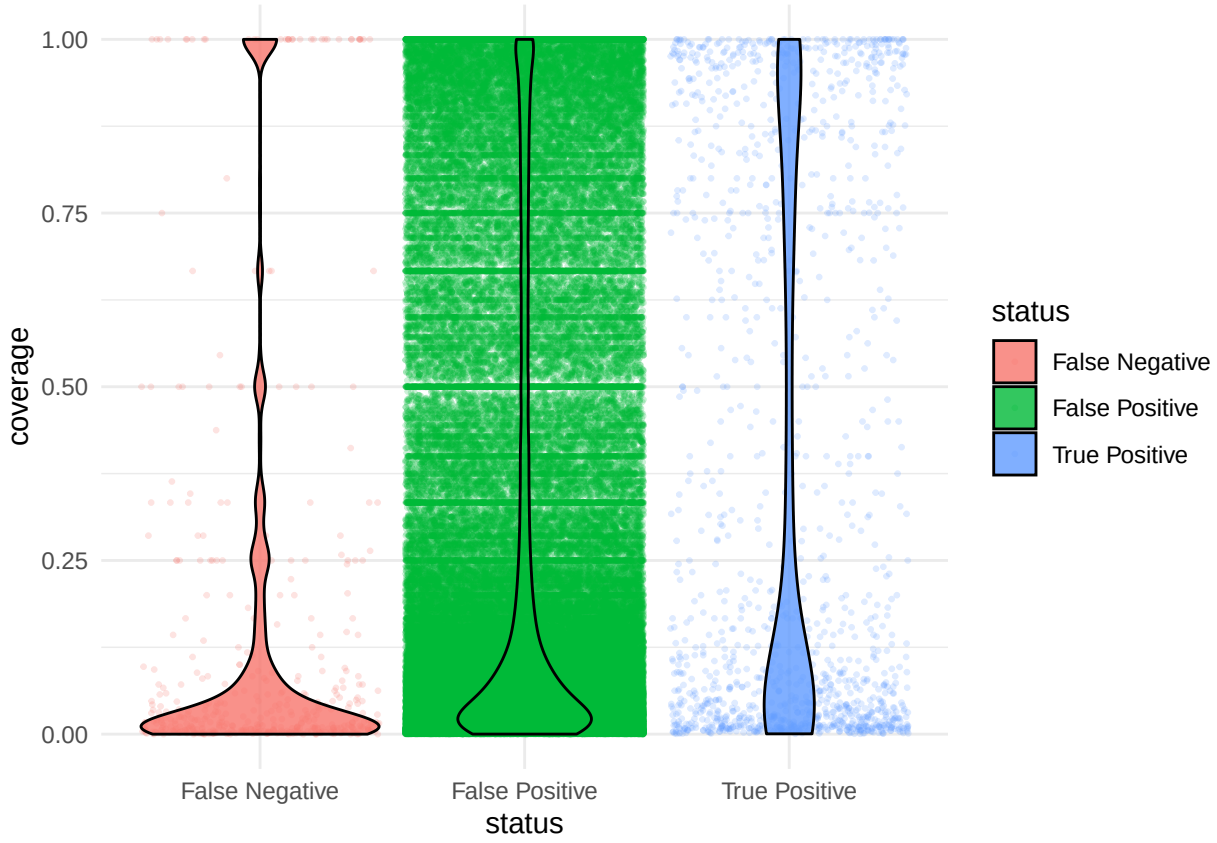
Looking at VCF details

```
## # A tibble: 3 x 7
##   status   pos_distinct distinct overall coverage supporting_reads overall_reads
## * <chr>           <int>    <int>   <int>    <dbl>            <dbl>         <dbl>
## 1 False N~          452      452     452  NA                   1            68
## 2 False P~       130767   130806  130813   0.0826              8           109
## 3 True Po~          674      674    1348   0.152               9           128.
```

```
coverage_plot <- ggplot(examine %>% filter(FILTER=="PASS" & status != "False Negative - GnomAD"),aes(sta

coverage_plot
```

```
## Warning: Removed 17 rows containing non-finite values (stat_ydensity).
```

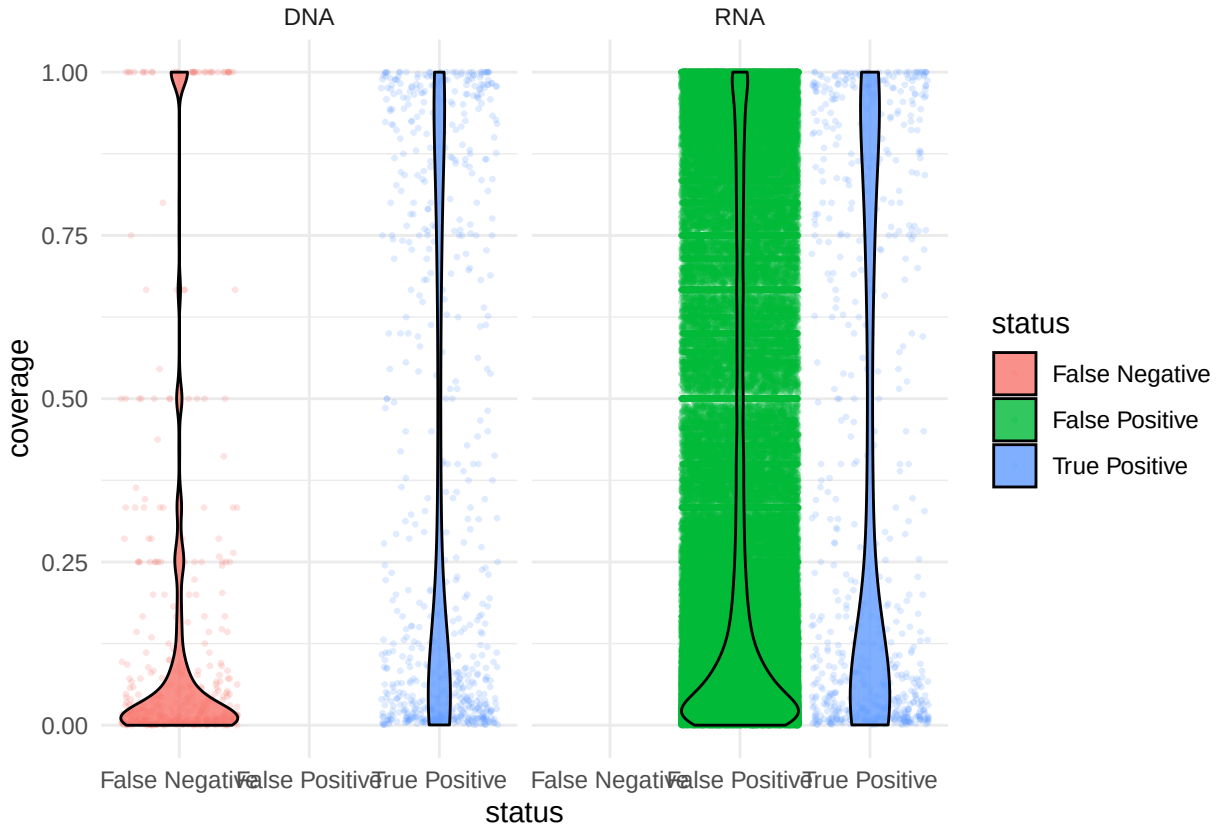
```
## Warning: Removed 17 rows containing missing values (geom_point).
```



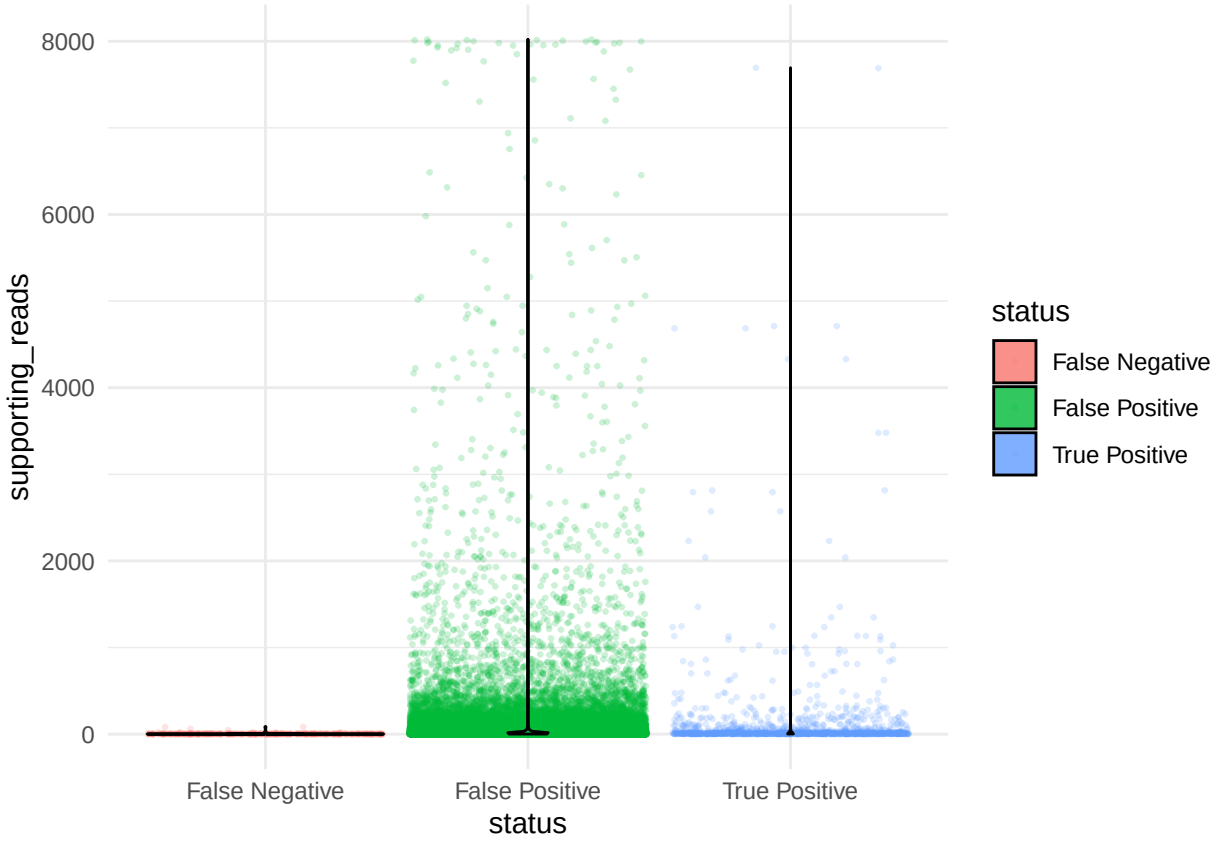

```
coverage_plot + facet_wrap(vars(src))
```

```
## Warning: Removed 17 rows containing non-finite values (stat_ydensity).
```

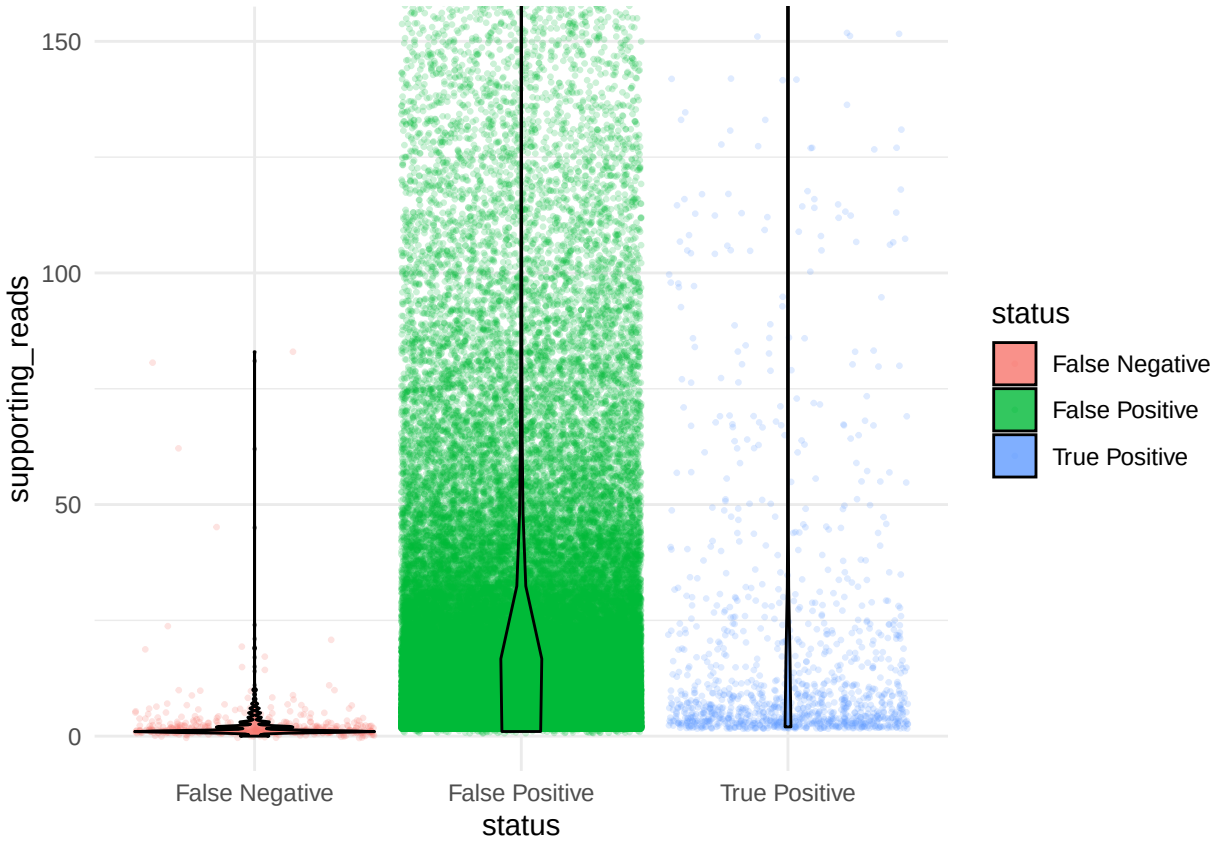
```
## Warning: Removed 17 rows containing missing values (geom_point).
```



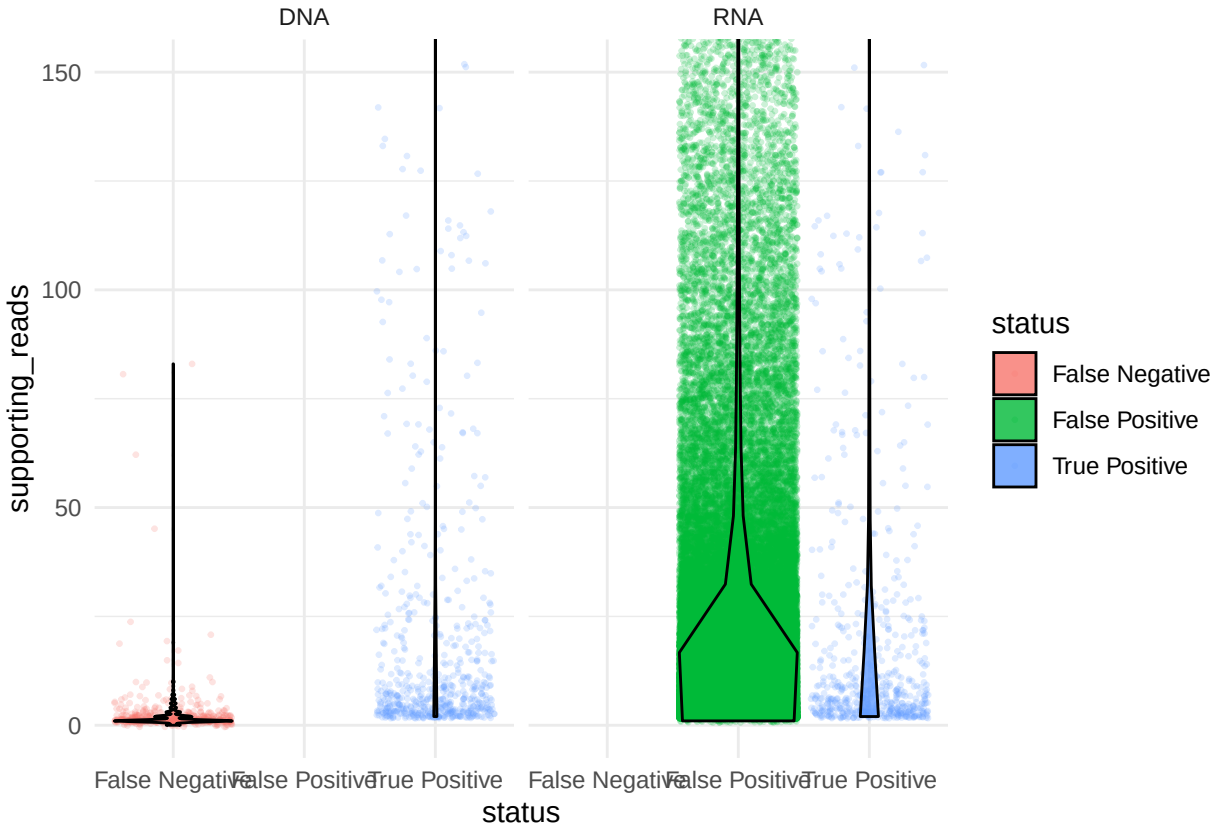

```
support_plot <- ggplot(examine %>% filter(FILTER=="PASS" & status != "False Negative - GnomAD"),aes(sta

support_plot
```



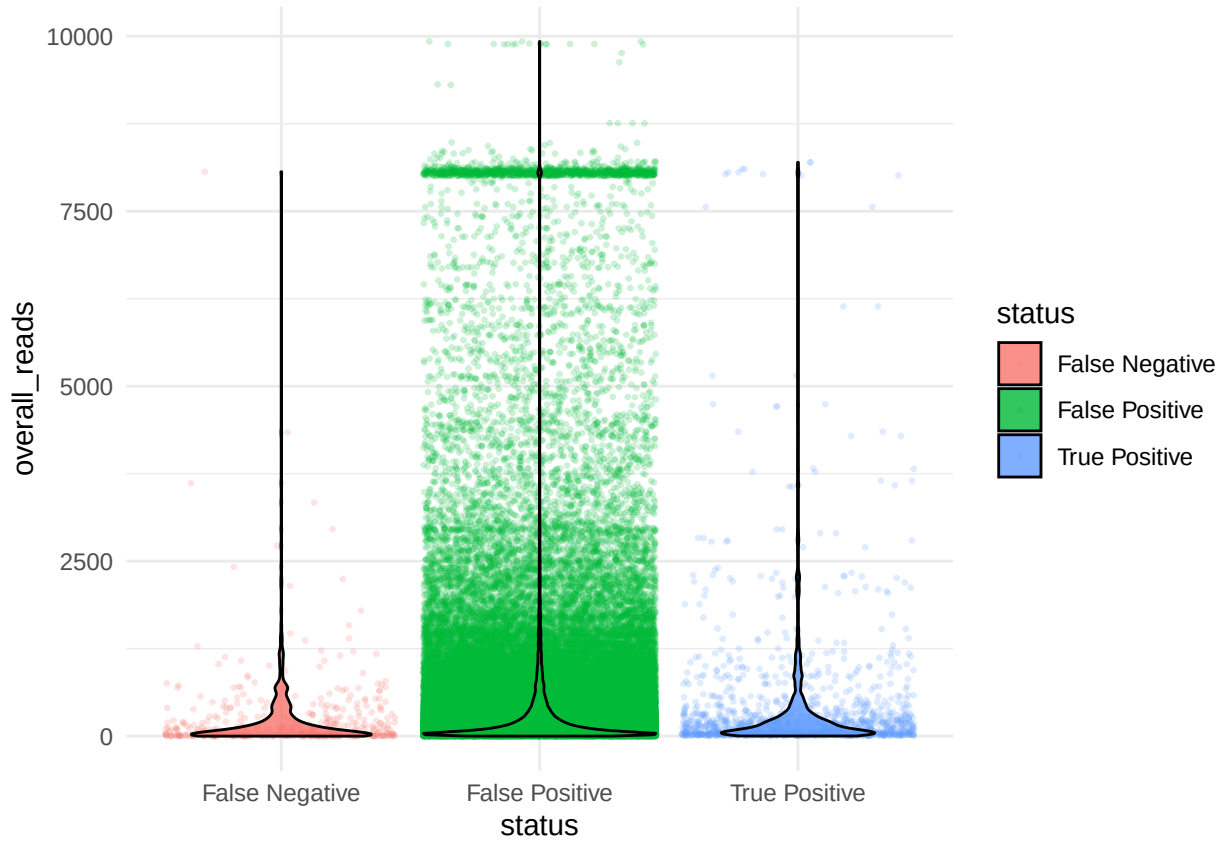

```
support_plot +  coord_cartesian(ylim = c(0, 150))
```



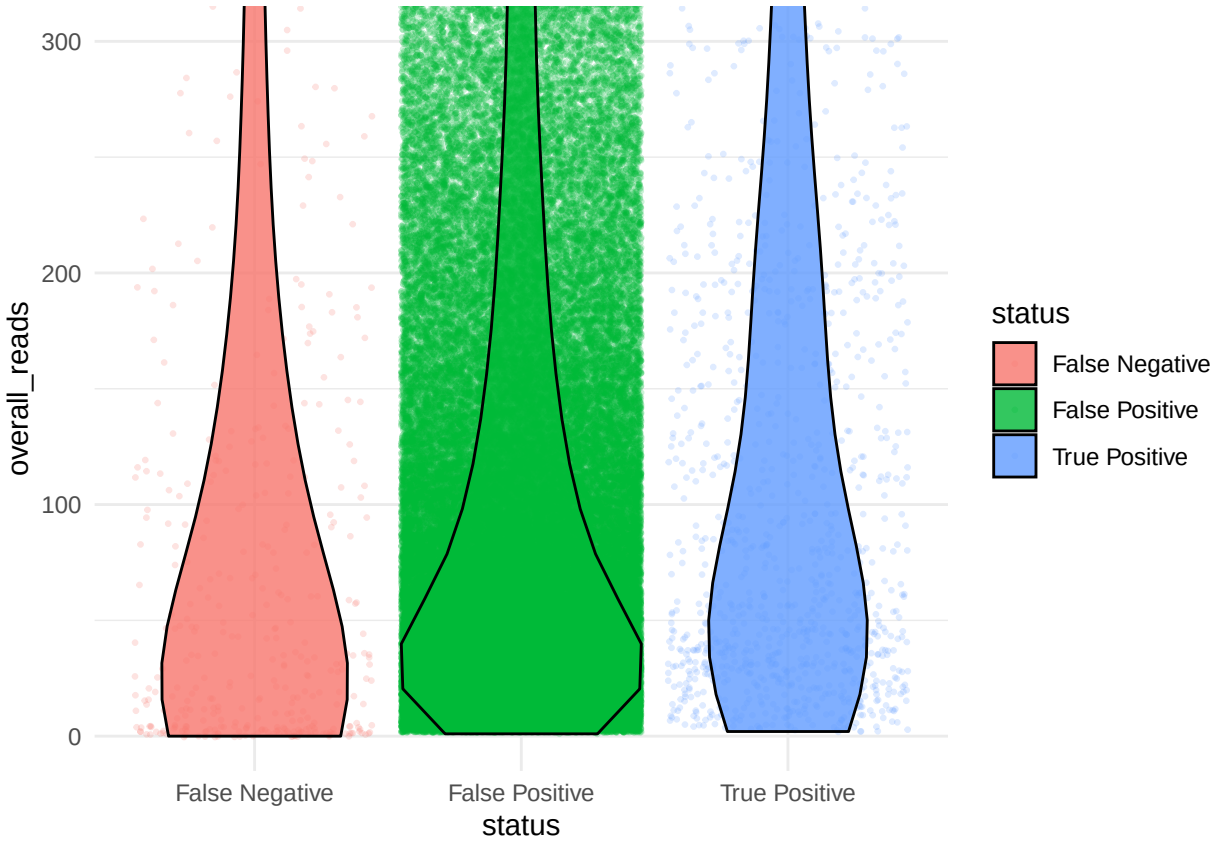

```
support_plot +  coord_cartesian(ylim = c(0, 150)) + facet_wrap(vars(src))
```



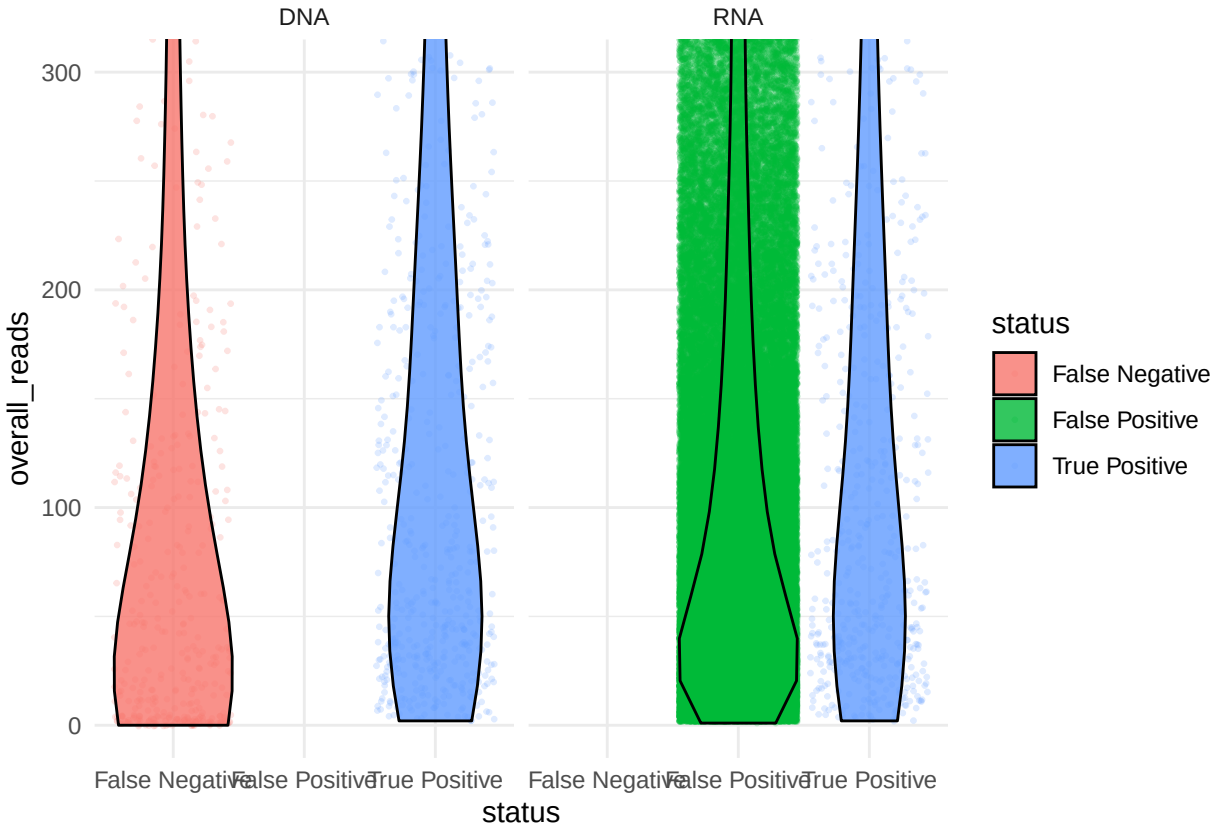

```
reads_plot <- ggplot(examine %>% filter(FILTER=="PASS" & status != "False Negative - GnomAD"),aes(statu

reads_plot
```




```
reads_plot + coord_cartesian(ylim = c(0, 300))
```

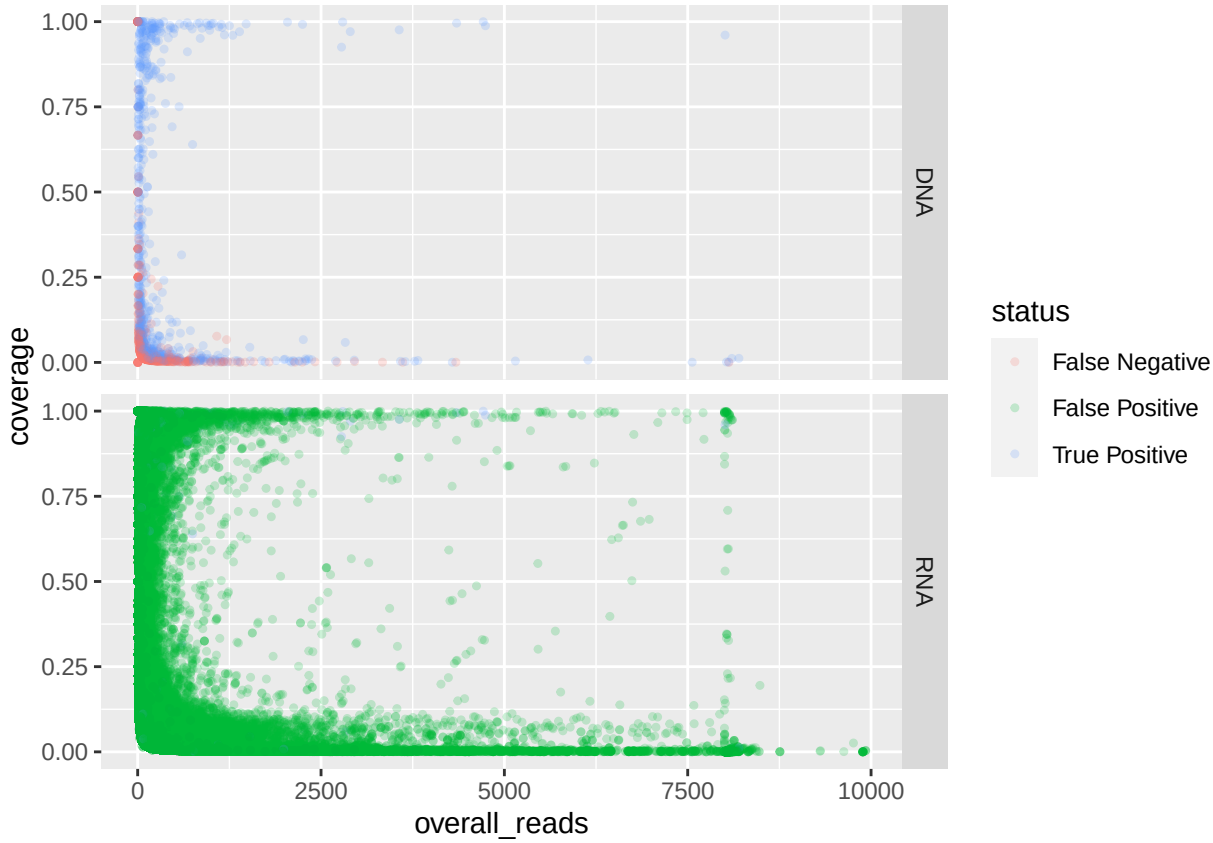



```
reads_plot + coord_cartesian(ylim = c(0, 300)) + facet_wrap(vars(src))
```




```
ggplot(examine %>% filter(FILTER=="PASS" & status != "False Negative - GnomAD"), aes(overall_reads, cov
```

```
## Warning: Removed 17 rows containing missing values (geom_point).
```



Looking at Coverage in Normal

Used freebayes on the normal BAM with the vcf file from the RNA and DNA run which will give depth of reads.

```
freebayes -f /scratch/drkthomp/protect-index/hg38.fa -r chr6 --report-monomorphic -4 -l -G 0 -C 0 -F 0
```

Right now can only do this analysis on chromosome 6, since I only grabbed the normals from there.

```
#gc()
memory.limit(size = 35000)
```

```
## [1] 35000
```

```
chr6_dna <- examine %>% filter(CHROM == "chr6" & src=="DNA")
chr6_rna <- examine %>% filter(CHROM == "chr6" & src=="RNA")


chr6_normal <- getVCF(paste0(file_loc, "/", "chr6_normal_forced"))
```

The problem is that I'm not sure what corresponds to "supporting reads" and "overall reads" or even "coverage" based on the descriptions. The possibilities:

ID	Description
NS	Number of samples with data
DP	Total read depth at the locus
DPB	Total read depth per bp at the locus; bases in reads overlapping / bases in haplotype
NUMALT	"Number of unique non-reference alleles in called genotypes at this position.
RO	Count of full observations of the reference haplotype.
AO	Count of full observations of this alternate haplotype.
technology.ILLUMINA	Fraction of observations supporting the alternate observed in reads from ILLUMINA

I would maybe process it something like this.

```
chr6_normal_touchup <- chr6_normal %>%
  filter(CHROM == "chr6") %>%
  rename(supporting_reads = AO, overall_reads = DPB) %>%
  mutate(supporting_reads = as.double(supporting_reads)) %>%
  mutate(overall_reads = as.double(overall_reads)) %>%
  mutate(coverage = as.double(supporting_reads/overall_reads)) %>%
  mutate(status = recode(POS, !!!deframe(chr6_dna %>% select(POS, status))))

# we don't actually care about the RNA
chr6_lookat <- bind_rows("DNA" = chr6_dna, "normal" = chr6_normal_touchup, .id ="src")

chr6_lookat %>% group_by(status,src) %>% summarise(supporting_reads=median(supporting_reads),overall_rea
```

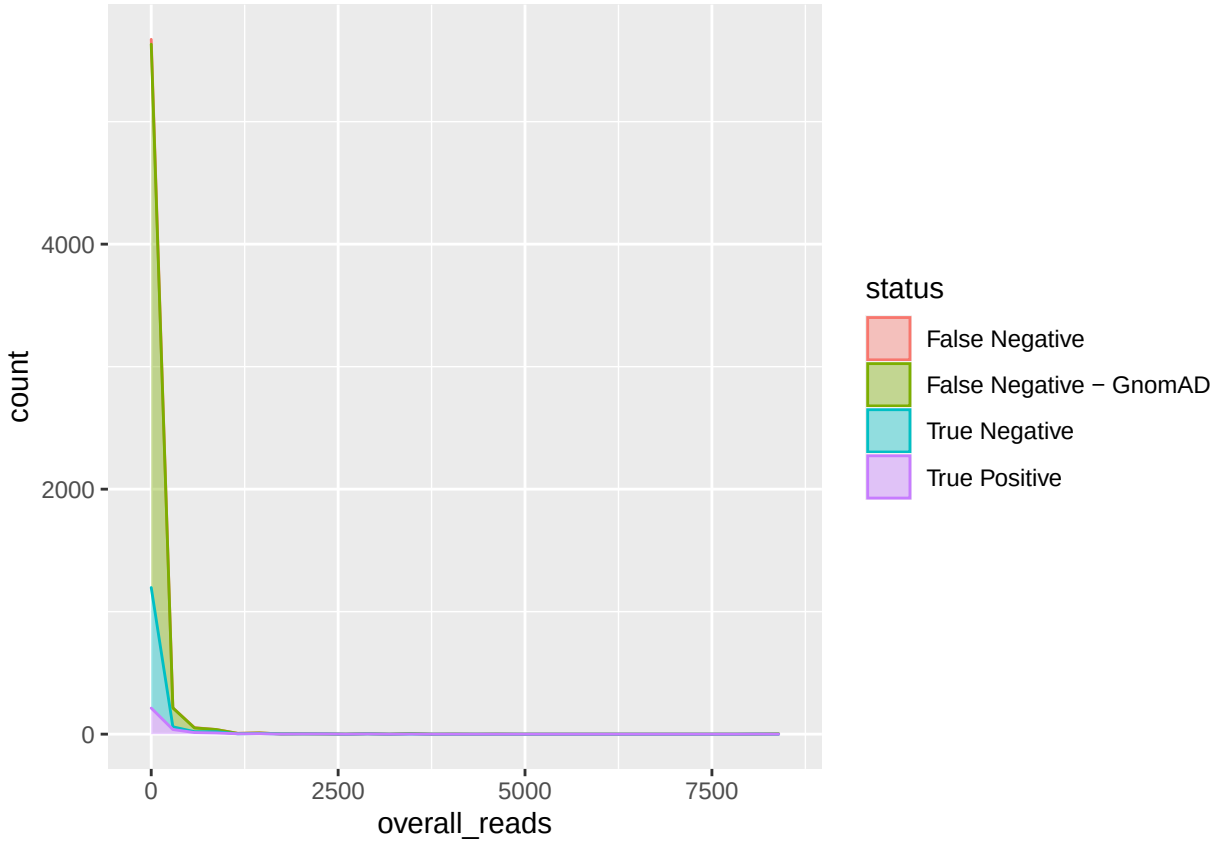
```
## `summarise()` has grouped output by 'status'. You can override using the `.groups` argument.
```

```
## # A tibble: 8 x 4
## # Groups:   status [4]
##   status                  src    supporting_reads overall_reads
##   <chr>                   <chr>             <dbl>         <dbl>
## 1 False Negative          DNA                   1            39
## 2 False Negative          normal                0            13
## 3 False Negative - GnomAD DNA                  NA            NA
## 4 False Negative - GnomAD normal               0             9
## 5 True Negative           DNA                  NA            NA
## 6 True Negative           normal                0            12
## 7 True Positive           DNA                  11           181
## 8 True Positive           normal                0            10
```

```
area <- ggplot(filter(chr6_lookat), aes(overall_reads, fill=status,color=status)) + geom_area(stat="bin
area
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

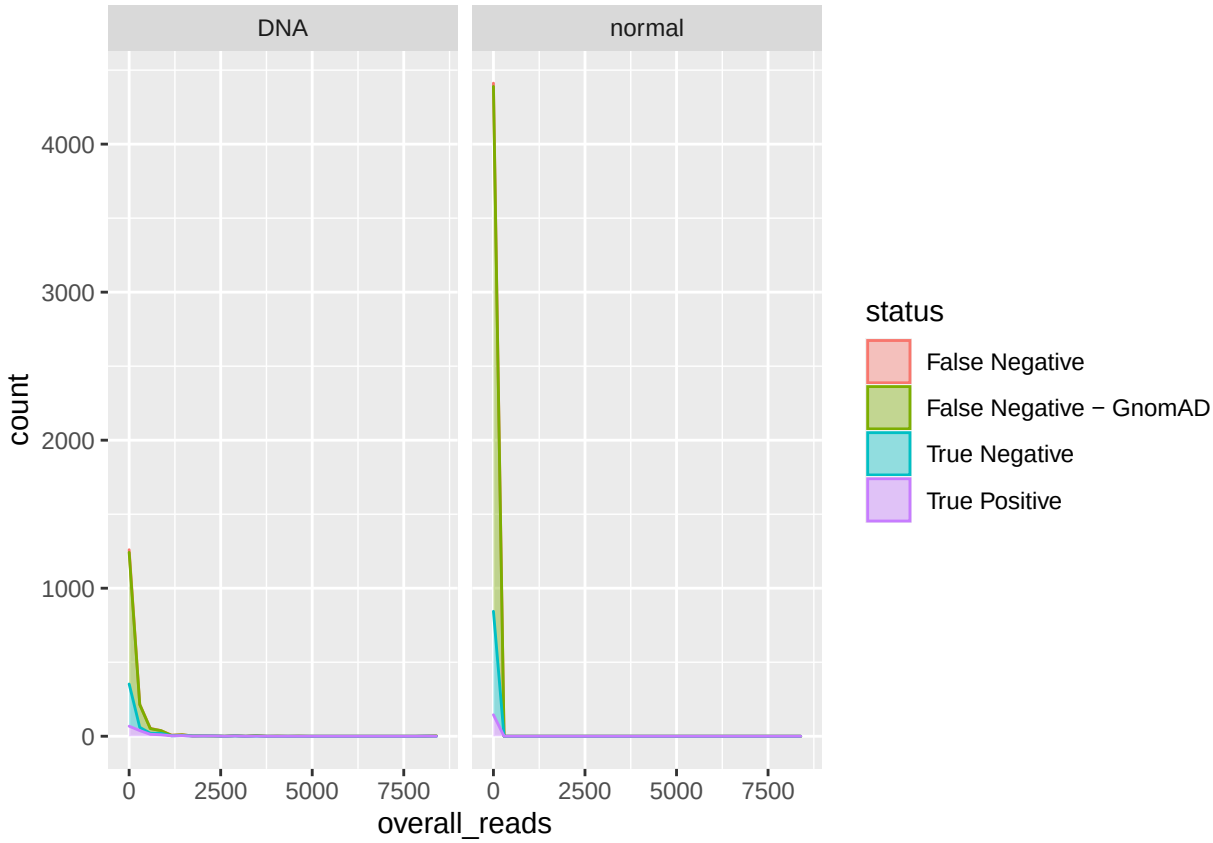
```
## Warning: Removed 2808 rows containing non-finite values (stat_bin).
```




```
area + facet_wrap(vars(src))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

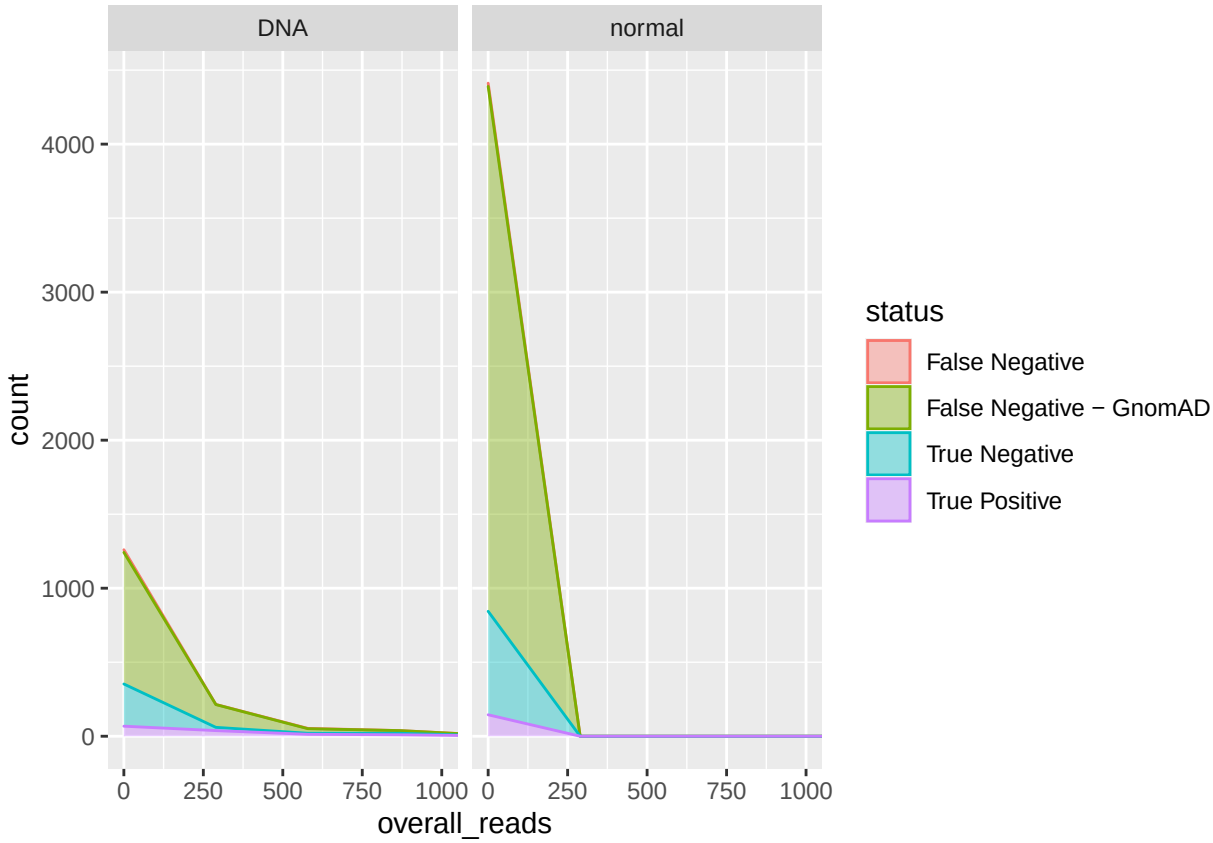
```
## Warning: Removed 2808 rows containing non-finite values (stat_bin).
```




```
area + facet_wrap(vars(src)) + coord_cartesian(xlim=c(0,1000))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

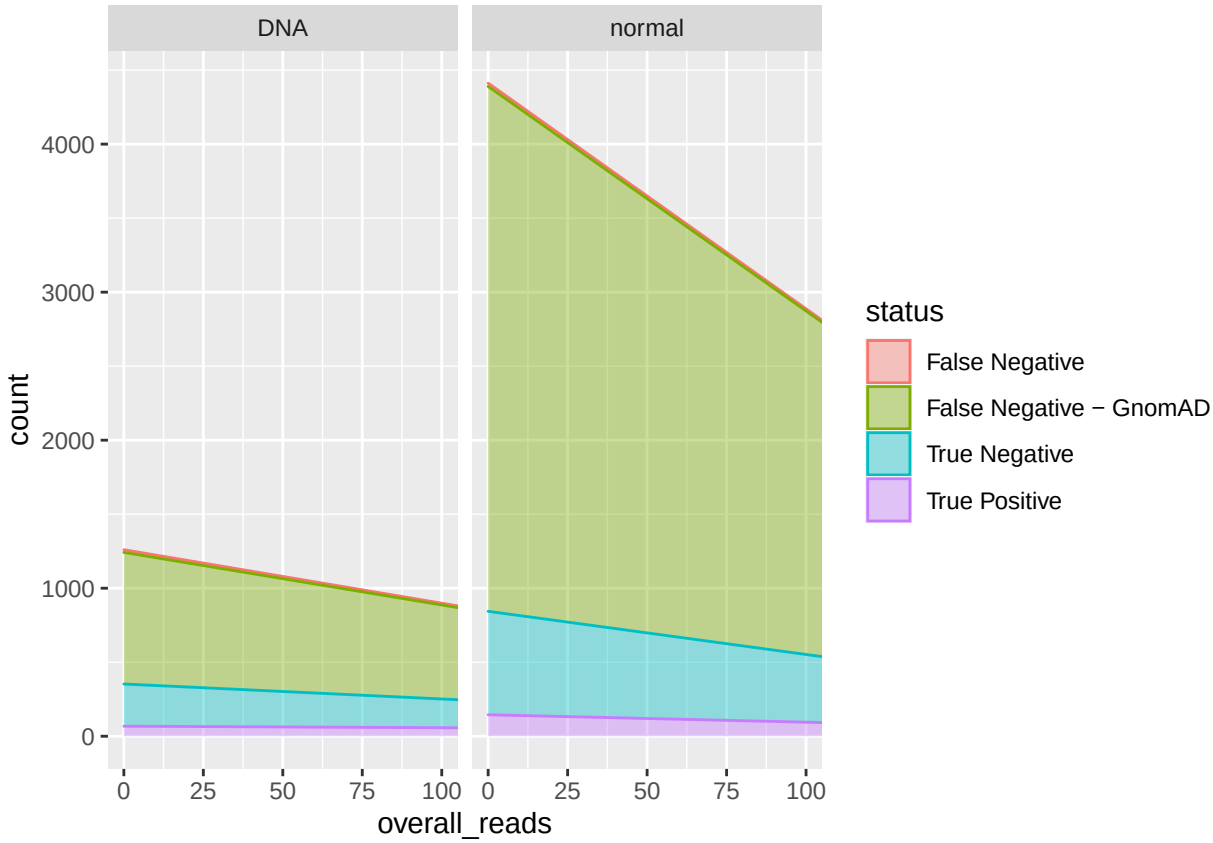
```
## Warning: Removed 2808 rows containing non-finite values (stat_bin).
```




```
area + facet_wrap(vars(src)) + coord_cartesian(xlim=c(0,100))
```

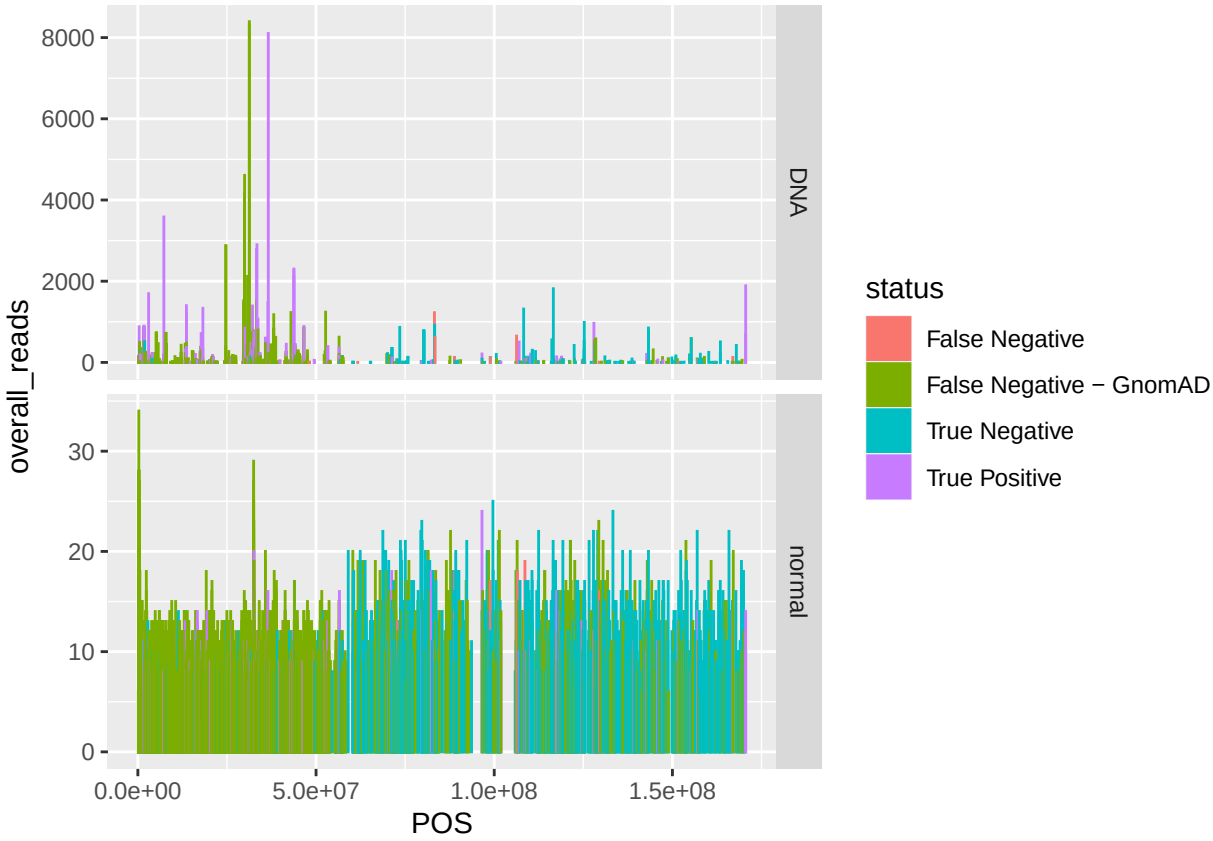
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 2808 rows containing non-finite values (stat_bin).
```



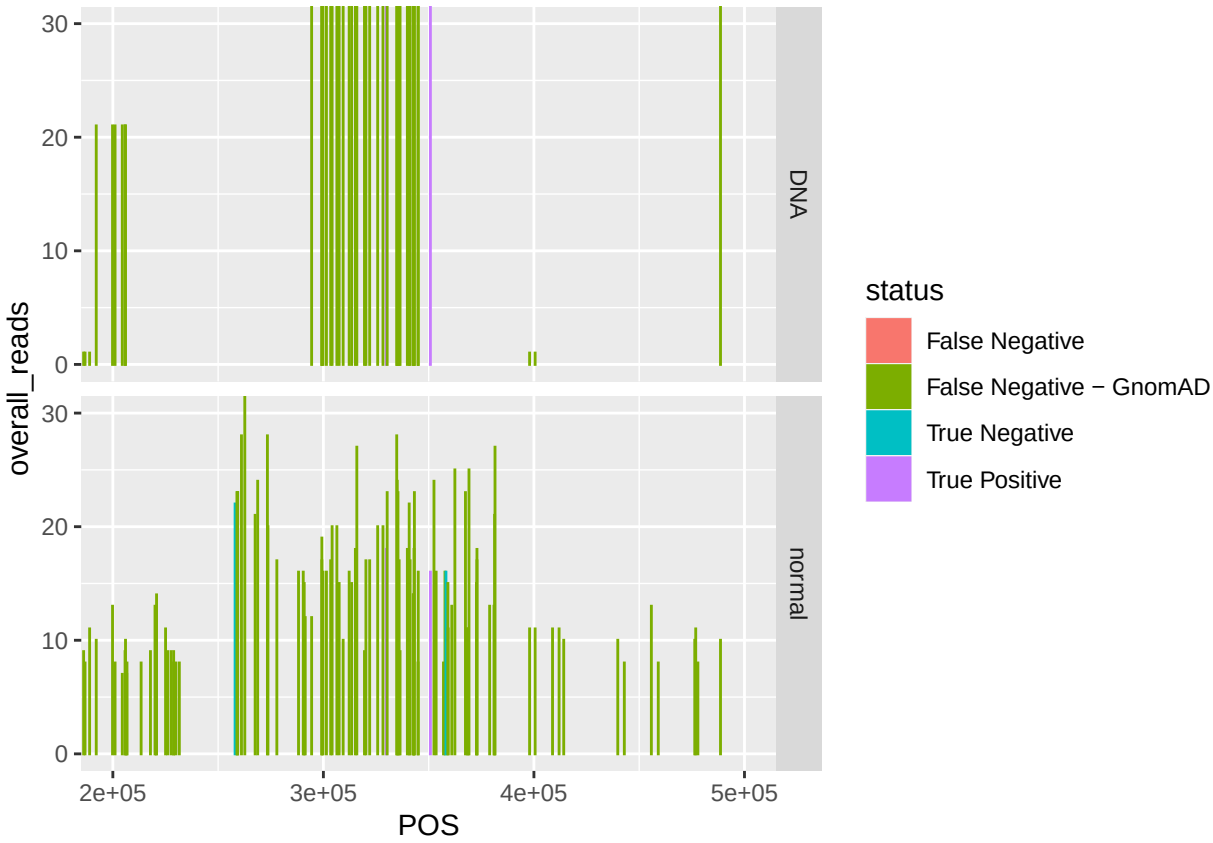

```
overall_look <- ggplot(chr6_lookat, aes(POS, overall_reads,color=status,fill=status)) + geom_col(positi
overall_look
```

```
## Warning: Removed 2808 rows containing missing values (geom_col).
```




```
overall_look +  coord_cartesian(xlim = c(200000, 500000),ylim=c(0,30))
```

```
## Warning: Removed 2808 rows containing missing values (geom_col).
```

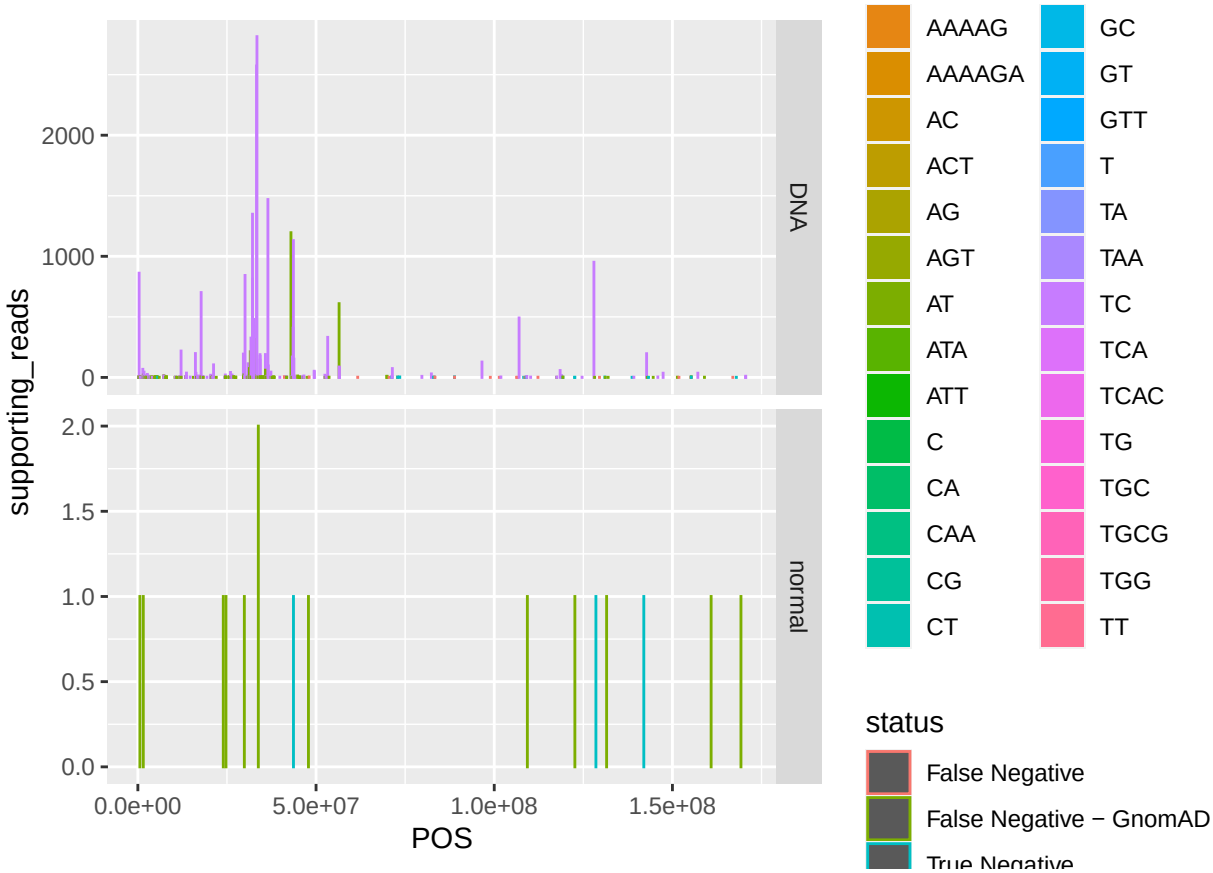



```
ggplot(chr6_lookat, aes(POS,overall_reads,color=src,fill=src)) + geom_col(position="dodge") + facet_gri
```

```
## Warning: Removed 2808 rows containing missing values (geom_col).
```


```
ggplot(chr6_lookat, aes(POS, supporting_reads,color=status,fill=ALT)) + geom_col(position="dodge") + fa
```

```
## Warning: Removed 2808 rows containing missing values (geom_col).
```



Likely we shouldn't consider something in the normal as known that has less than ten overall reads.

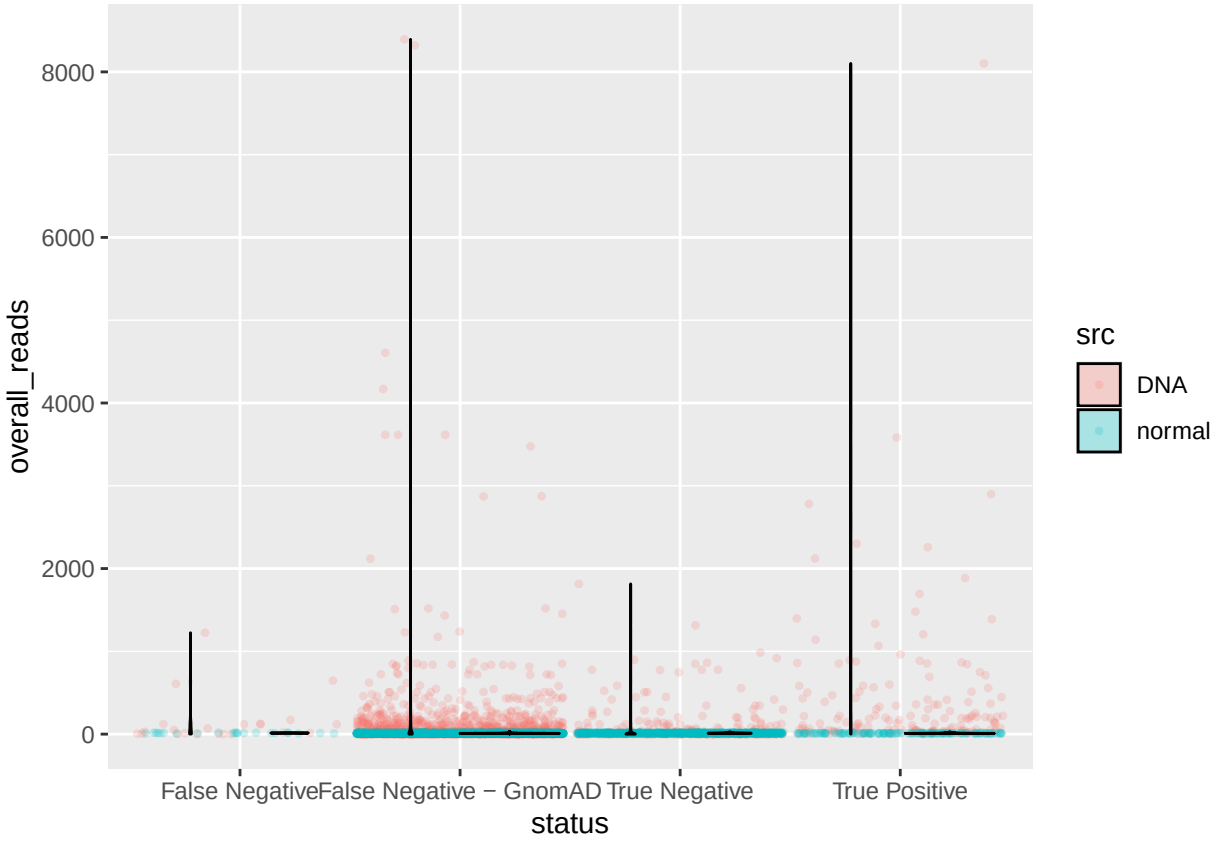
```
chr6_compare <- inner_join(chr6_normal_touchup,chr6_dna,by="mut_id",suffix=c("n", "d"))
chr6_compare %>% group_by(statusd) %>% summarise(dna_supporting_reads=median(supporting_readsd),dna_ove
```

```
## # A tibble: 4 x 5
##   statusd          dna_supporting_rea~ dna_overall_rea~ normal_overall_re~ count
## * <chr>                          <dbl>            <dbl>               <dbl> <int>
## 1 False Negative                     1               49                  13    21
## 2 False Negative ~                  NA               NA                   9  3517
## 3 True Negative                     NA               NA                  12   694
## 4 True Positive                     11              181                  10   145
```

```
read_distr <- ggplot(chr6_lookat, aes(status,overall_reads,fill=src,color=src)) + geom_jitter(alpha=0.2
read_distr
```

```
## Warning: Removed 2808 rows containing non-finite values (stat_ydensity).
```

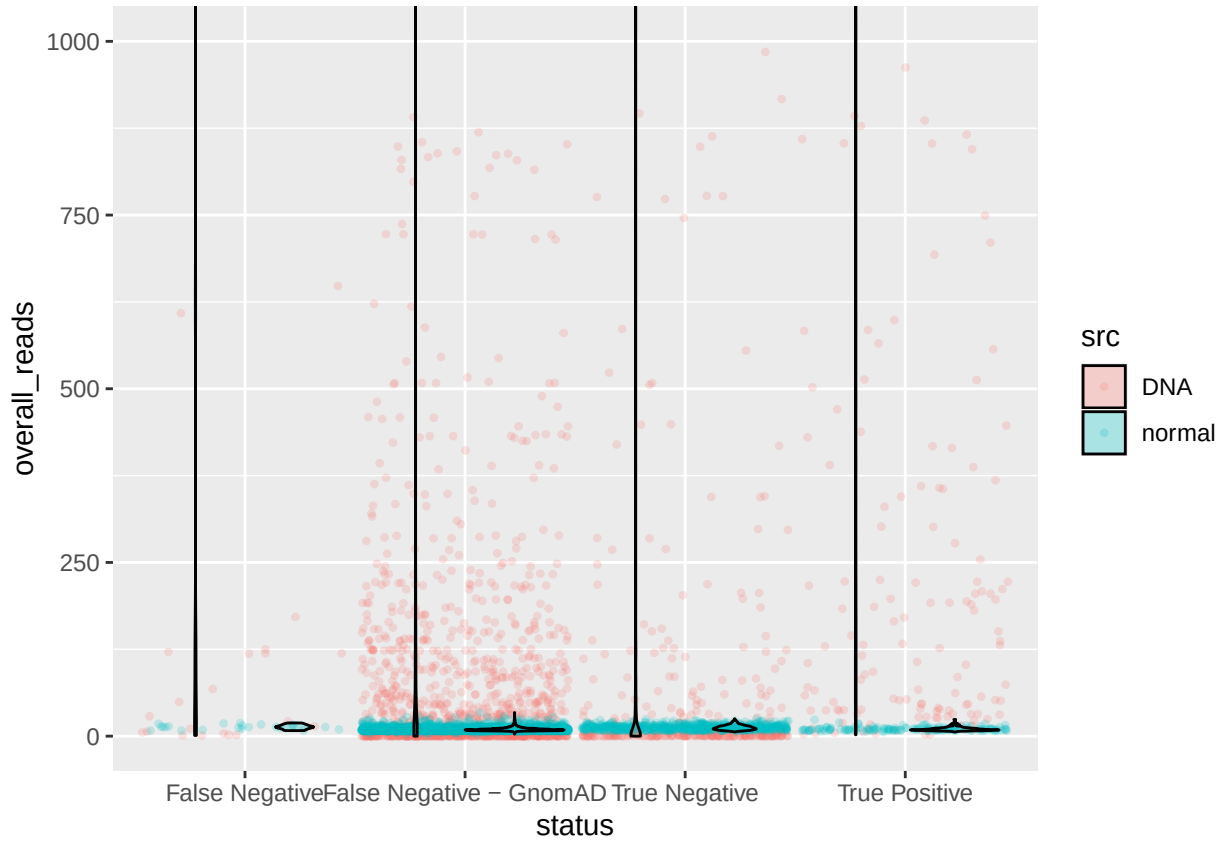
```
## Warning: Removed 2808 rows containing missing values (geom_point).
```




```
read_distr + coord_cartesian(ylim=c(0,1000))
```

```
## Warning: Removed 2808 rows containing non-finite values (stat_ydensity).
```

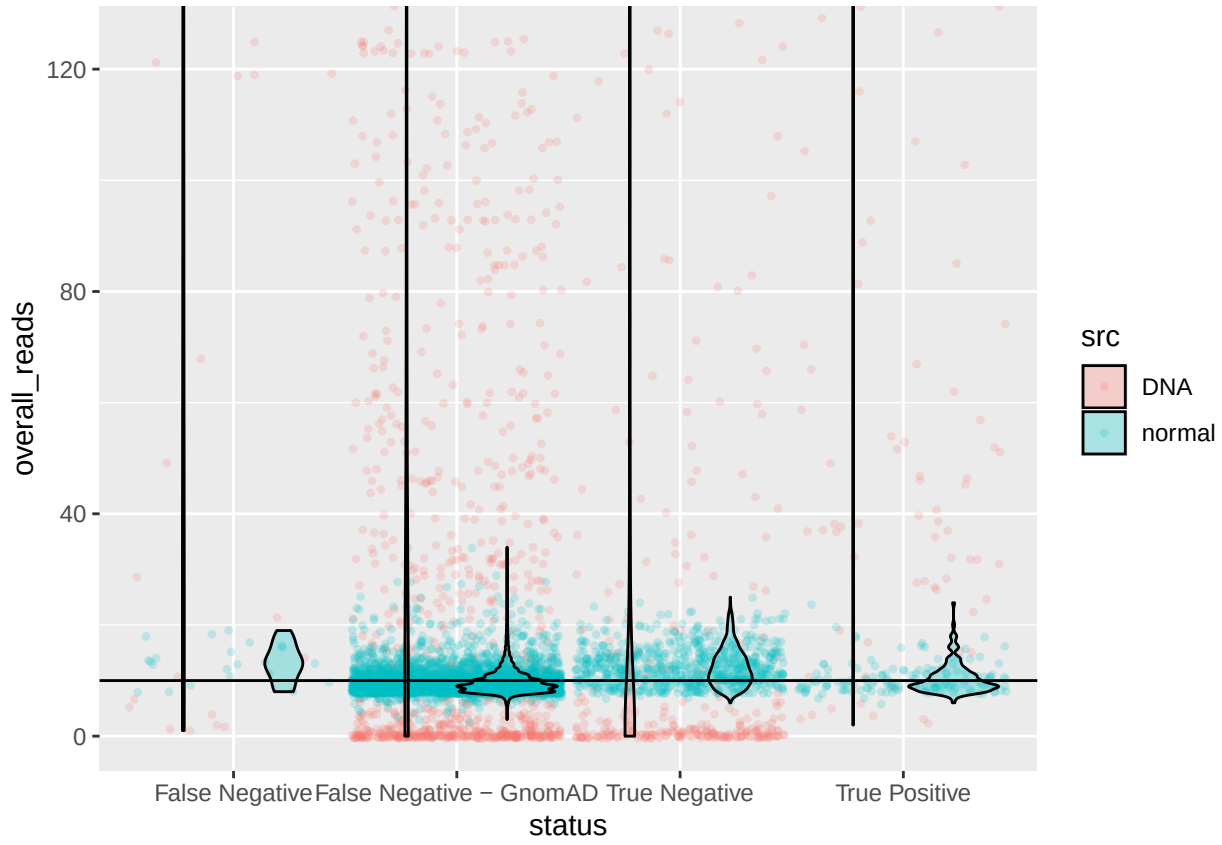
```
## Warning: Removed 2808 rows containing missing values (geom_point).
```




```
read_distr + coord_cartesian(ylim=c(0,125)) + geom_hline(yintercept=10)
```

```
## Warning: Removed 2808 rows containing non-finite values (stat_ydensity).
```

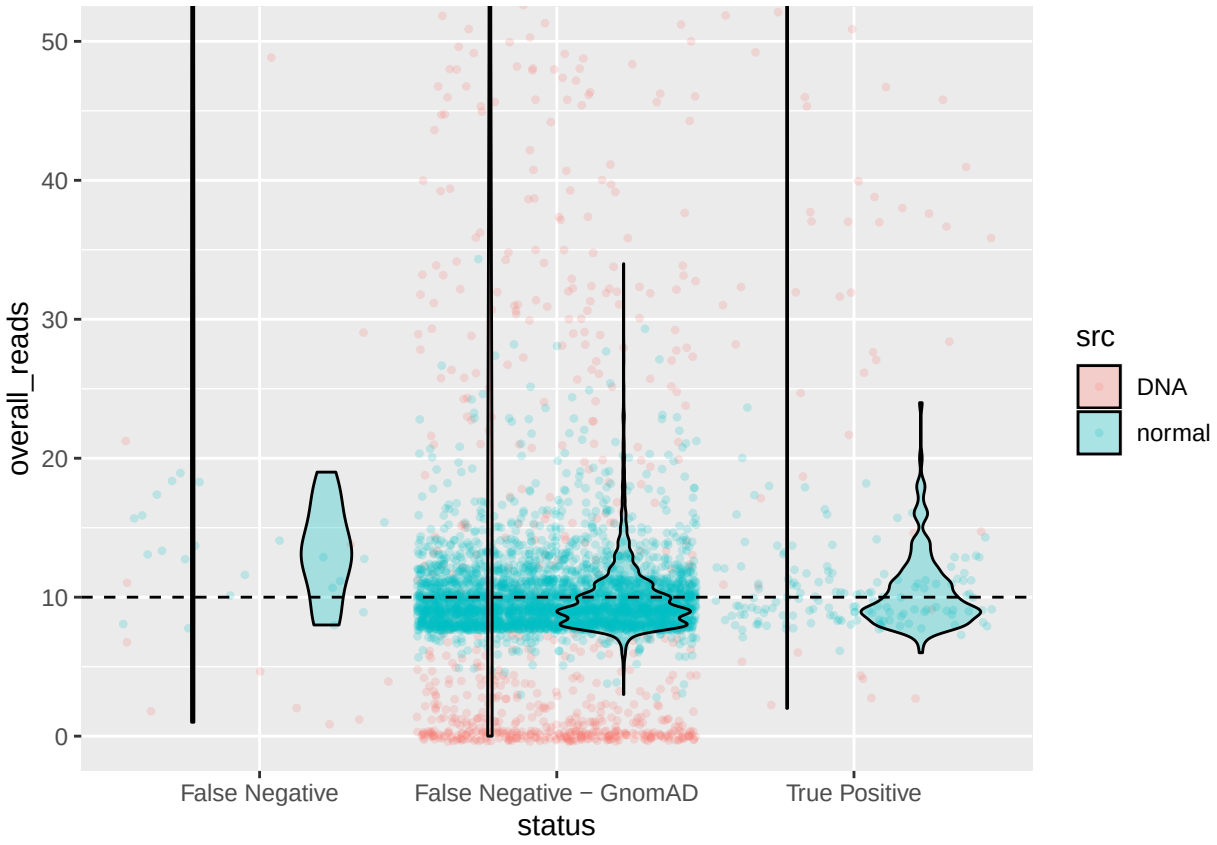
```
## Warning: Removed 2808 rows containing missing values (geom_point).
```




```
ggplot(chr6_lookat %>% filter(status != "True Negative"), aes(status,overall_reads,fill=src,color=src))
```

```
## Warning: Removed 2435 rows containing non-finite values (stat_ydensity).
```

```
## Warning: Removed 2435 rows containing missing values (geom_point).
```



2 - Examining Binding Predictions

```
getBinding = function(base, fileType) {
  fileName <-
    paste0("~/Documents/igv-analysis/all_chromosomes/binding/chr", base,
           "_", fileType, ".list")
  if (file.exists(fileName)) {
    binding <- read.delim(fileName, header = FALSE)
    names(binding) <-
      c(
        "allele",
        "pept",
        "normal_pept",
        "pname",
        "core",
        "zero",
        "tumor_pred",
        "normal_pred",
        "ENSG",
        "gene",
        "ENST"
      )
    return( separate_rows(arrange_all(binding),ENST, sep=","))
  }
```

```
}

compareClass = function(mhcClass) {
  chroms <- c(1,2,3,4,5,7,8,9)
  toRun <- parse(text = paste0("getBinding('", chroms, "', '", mhcClass, "')"))
  header <- paste0("binding", mhcClass, ".")
  for (i in seq_along(toRun))
    assign(paste0(header, "chr", chroms), eval(toRun[[i]]))
  fullList <- mget(ls(pattern = paste0('^', header, '.*')))
  return(fullList)
}

closeCompareBinding = function(fullList, comparison, mhcClass){
    uniqueMHC <- fullList
    for( i in seq_along(uniqueMHC)){
      uniqueMHC[i] <- unique(uniqueMHC[[i]][comparison])
    }
    test <- euler(uniqueMHC, shape = "ellipse")
    grid.arrange(grobs = list(
      plot(euler(uniqueMHC, shape = "ellipse"), quantities = TRUE, legend = TRUE )),
      top = paste(mhcClass, comparison, "Comparison"))
    baseMHC <- uniqueMHC[[paste0("binding", mhcClass, ".", baseline)]]
    compMHC <- uniqueMHC[[paste0("binding", mhcClass, ".", mainCompare)]]
    writeLines("\n#### Difference\n")
    writeLines(setdiff(baseMHC, compMHC))
    writeLines("\n#### Intersect\n")
    writeLines(intersect(baseMHC,compMHC))
  }
```

MHCI

```
mhcClass <- 'mhci'
mhciBindingList <- compareClass(mhcClass)
mhciBindingList %>%
    bind_rows(.id = "patients") %>%
    mutate(patients = str_remove_all(patients, paste0("binding", mhcClass, "."))) %>%
    mutate(patients = str_remove_all(patients, "_chr6")) %>%
    mutate(patients = str_remove_all(patients, "variants_")) %>% group_by(patients) %>%
  summarise(
    number = n(),
    tumor_pred = mean(tumor_pred),
    normal_pred = mean(normal_pred, na.rm = TRUE),
    diff = mean(tumor_pred - normal_pred)
  )
```

Gene

```
closeCompareBinding(mhciBindingList, 'gene', mhcClass)
```

ENST

```
closeCompareBinding(mhciBindingList, 'ENST', mhcClass)
```

MHCII

```
mhcClass <- 'mhcii'
mhciiBindingList <- compareClass(mhcClass)
mhciiBindingList %>%
    bind_rows(.id = "patients") %>%
    mutate(patients = str_remove_all(patients, paste0("binding", mhcClass, "."))) %>%
    mutate(patients = str_remove_all(patients, "_chr6")) %>%
    mutate(patients = str_remove_all(patients, "variants_")) %>% group_by(patients) %>% summarise(
    number = n(),
    tumor_pred = mean(tumor_pred),
    normal_pred = mean(normal_pred, na.rm = TRUE),
    diff = mean(tumor_pred - normal_pred)
  )
```

Gene

```
closeCompareBinding(mhciiBindingList, 'gene', mhcClass)
```

ENST

```
closeCompareBinding(mhciiBindingList, 'ENST', mhcClass)
```

3 - Examining Rankboost